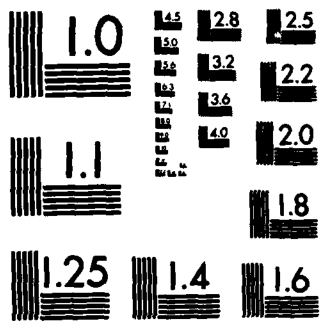



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A144 562

DTIC FILE COPY

PERFORMANCE IMPROVEMENTS OF THE
PHONEME RECOGNITION ALGORITHM

THESIS

James E. Fletcher
Captain, USAF

AFIT/GE/EE/84J-2





DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

84 08 20 090

1



PERFORMANCE IMPROVEMENTS OF THE
PHONEME RECOGNITION ALGORITHM

THESIS

James E. Fletcher
Captain, USAF

AFIT/GE/EE/84J-2

AFIT/GE/EE/84J-2

# PERFORMANCE IMPROVEMENTS OF THE PHONEME RECOGNITION ALGORITHM

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering



James E. Fletcher, B.S.
Captain, USAF

June 1984

Approved for public release; distribution unlimited

## Preface

This work has been motivated by the work of Lt. Karl Seelandt, December 1981 graduate of Air Force Institute of Technology and the research of Maj. Larry Kizer and Dr. Matthew Kabrisky, professors of Electrical Engineering, Air Force Institute of Technology. This research used Seelandt's phonemes as a basis for determining an optimum length of the phonemes. In addition, speech synthesis was attempted using single vector or time slice phonemes.

I would like to thank my advisors Maj. Larry Kizer and Dr. Matthew Kabrisky for their patience and guidance during this project.

I would also like to thank my wife, Susan, for her help as a partner during my work. She has given me moral support during my research and without her contributions this project could never have been completed.

## Contents

## List of Figures

# List of Tables

AFIT/GE/EE/84M-2

## Abstract

An analysis of speech is made by comparing single, two and three time slice phonemes to five time slice phonemes (.08 sec/time slice). All phonemes were created from the same speech with the single, two and three time slice phonemes created from portions of the five time slice phonemes. It was found that the single time slice phonemes compared favorably with the five time slice phonemes in recognizing speech only if the frequency components remained relatively constant over a period of time, such as the vowel and nasal sounds. The two time slice phonemes showed results that began to duplicate those of the five time slice phonemes, but still had inconsistent results identifying fricative sounds. Three time slice phonemes results showed a closer correlation with the results of the five time slice phonemes than those of the one and two time slice phonemes. All results were obtained using a 64 point sampled, Hamming windowed, Discrete Fourier Transform. The recognition results for each time slice of speech, using various length phonemes, are tabulated and the results are used to re-synthesize the original speech. This was done by using digitized speech composed of the middle time slices from the 71 five time slice phonemes.

Results indicated that the synthesized speech was understandable when the recognition results successfully identified the proper phoneme for approximately 4 consecutive time slices. An extraneous phoneme choice in a consecutive grouping of a phoneme choice did not seriously degrade the output since it accounted for only an .08 second time slice.

# PERFORMANCE IMPROVEMENTS OF THE PHONEME RECOGNITION ALGORITHM

## I. Introduction

Speech recognition has been studied since the 1940's, where experiments were made with visual hearing. Subjects in this experiment studied spectrograms and interpreted what had been said (Ref 1).

In the 1950's automatic recognition was introduced in a paper that described a method of recognizing the utterance of digits (Ref 1:44-45).

### Justification

The need for automatic speech recognition stems from the fact that as the technology advances in information systems, a more direct interface is needed between man and machine. For military applications, this fact has come to the attention of engineers designing information systems for future aircraft. In particular, this problem is being investigated for the AFTI F-16 aircraft. In a combat environment such as a night attack mission, the pilot is required to monitor and operate navigation, terrain following sensors, and weapons delivery systems, in addition to monitoring the normal flight instruments. He's also required to make decisions concerning weapons delivery tactics, threat avoidance, and navigate to and from target areas. With such an increase in workload, flight safety and mission integrity are severely jeopardized utilizing present technology for the man-machine interface.

Utilizing both automatic speech recognition and speech synthesizing can further the development of narrow bandwidth voice communication. This can prove valuable where low frequency and very low frequency communications are used along with their characteristically narrow bandwidths.

Background

At the present time, there are commercially available speech recognition machines that perform limited speech recognition. There are still major drawbacks to these available machines.

First, most of these machines still have to go through a training session where a speaker recites the words the machine is to recognize. Even though the training period has been made fairly painless to the user, it still severely limits the vocabulary the machine can understand.

Second, because of the training required by the machines, they still are largely speaker dependent either for an individual or a small group of people. Drastic changes in a speaker's inflection could also degrade the recognition results even though the machine was trained by that individual.

Finally, recognition has been targeted mainly at the recognition of the utterance of single words. Problems with recognizing connected speech must also be addressed. This can prove to be more difficult because the utterance of an individual word is distorted by the proximity of other words (Ref 2:4).

Seelandt, in his thesis (Ref 2), approached the problem of recognizing connected speech. A phoneme recognition scheme was used with a set of 71 different phoneme templates used as the basis of his recognition routine. Phonemes were selected from connected speech that has been digitized by sampling the analog speech at 8000 HZ. The speech consisted of a series of utterances of the digits zero through nine. The selected phonemes represented 40 msec of speech or 5 time slices where, each time slice is 8 msec. The phonemes were chosen from Seelandt's own speech which consisted of four separate speechfiles. Phonemes were chosen with the aid of a software tool that allowed the user to observe, on the Tektronix 4010 graphic terminal, the spectrogram of 20 blocks or 800 msec portions of a speechfile and be able to simultaneously listen to any portion of the 20 blocks chosen. A block of speech can also be described as a group of 4 vectors. In this research project, the term vector and time slice are used interchangably with each vector containing 32 frequency components ranging from D. C. to 4000 HZ.

## Method

As in Seelandt's thesis, (Ref 2) the phoneme recognition scheme will use as a basis the 71 phonemes from Seelandt's speechfile. The recognition routines that are used were developed by Seelandt (Ref 2) and modified by Montgomery (Ref 3).

The basis for this recognition routine can be found in Potter, Kopp and Green's research where they demonstrated that after an education process, humans can recognize phonetic patterns in speech spectrograms (Ref 1). Using this as justification the premise is made here that a machine should be able to use the spectrogram data to recognize speech.

The problem to be addressed in this thesis is how much information in the phoneme prototype is needed to get the results Seelandt obtained with his prototype and how much information is needed to synthesize intelligable speech from the recognition routines. Information, as used here, refers to the number of time slices used in the phoneme prototypes.

## Scope

In Seelandt's thesis, the length of the phoneme prototypes was rather arbitrarily chosen to be 5 vectors. The purpose of this project was to determine if this was indeed an optimum length for the prototype and what, if any, were the performance differences of prototypes made up of fewer vectors. Information and software tools from Seelandt's thesis were used extensively since this was to be a follow-on study using his results.

Because there is no perfect model to compare recognition results with, the 5 vector prototypes used

by Seelandt would be used as a data base for comparison.
Other parameters were also to be kept constant throughout the project. These included using 6dB preemphasis beginning at 500 Hz, using the threshold value of 10 (See Seelandt, Ref 2) for the RMS energy in a speechfile and using a Hamming window in the computation of the spectrograms.

As shown in Seelandt's thesis (Ref 2), there was no performance advantage when a Hamming window was used instead of a rectangular window. With this in mind, the Hamming window was used only for the sake of consistancy.

Sequence of Presentation

The sequence to be followed in this report will be to first discuss the phoneme template development; secondly how qualitative analysis was accomplished, the next section will follow with how speech synthesis was accomplished, the fourth major topic will be how Martin's speech recognition routine was used in this project and last, the results and recommendations will be addressed.

## II. Phoneme Template Development

Previous work by Felkey (Ref 4) and Seelandt (Ref 2) led to the development of some of the software tools that were used in this effort.

PICTALK allows the user to input a file of processed speech through the Eclipse background and have it print a spectrogram on the Tektronix 4010 terminal that interfaces with the Eclipse foreground. Once the spectrogram has been printed, the program allows the user to select and listen to any portion of that speechfile. In order to output speech through the Crown amplifier, MONITR must be running simultaneously on the Nova. MONITR monitors the Interprocessor Buffer (IPB) and passes parameters from PICTALK that is used by the Channel Operating Sustem (CHOPS).

PHOGEN is used to generate the prototype set of phonemes. These prototypes are 5 time slices long and the user has the option to either create a new set of prototypes or create an averaged set of prototypes. The source for the prototypes are the speech file. When averaging is done, the following formula is used:

$$z(i) = \frac{(x(i)*n) + y(i)}{n + 1}$$

where: n is the number of times averaging has occured

y(i) is the ith component of a time slice to be averaged in.

X(i) is the ith component of a time slice from the existing prototype

z(i) is the ith component of a time slice from the new prototype (Ref 2)

Problems encountered

In general, problems were encountered when three of Seelandt's programs were compiled and run.

When PICTALK and MONITR were run, a Nova error with value 163 occurred when an attempt was made to listen to a portion of speech. In order to try and pinpoint what caused the error, flags were used and values passed to subroutines were echoed. By observing the flags and backtracking through the code by hand, it was found that the error originated from the subroutine CHANNEL. CHANNEL is a Fortran subroutine designed to interface programs running on the Nova to the Cromemeo Z-2 computer with the channel operating system loaded and running in it.

Studying the documentation on CHANNEL revealed that the Nova error was equivalent to channel error 35 which meant that the file to be transferred was not evenly divisable by the channel block count. This was very puzzling since the number of blocks in the file to be transferred was set as a constant of 86 and the channel block count was a constant of 2. The only explanation for the error was that the file used was actually less than 86 blocks and if this occurs, the task is aborted. This was corrected by making the blocks transferred a user defined variable instead of a constant.

Another problem encountered using PICTALK and MONITR was the fact that it needed two terminals on the Eclipse and one terminal on the Nova in order to operate. Initially, PICTALK was modified so that only one Eclipse terminal was needed and the modified program was renamed TEKTALK. Later, a second modification was made after the A/D/A converter for the Eclipse was operating. With this modification all the operations carred out by the two computers were consolidated into TEKTALK and a swapped program, TALK. This allowed the Eclipse to perform the entire operation of displaying the spectrogram and output the desired portions of the speechfile through the Crown amplifiers and speakers in the same manner as PICTALK and MONITR (Ref 5). This also allowed the entire operation to be executed on one terminal. Figure 1 illustrates the configuration for operating the Eclipse A/D/A converter with TEKTALK.

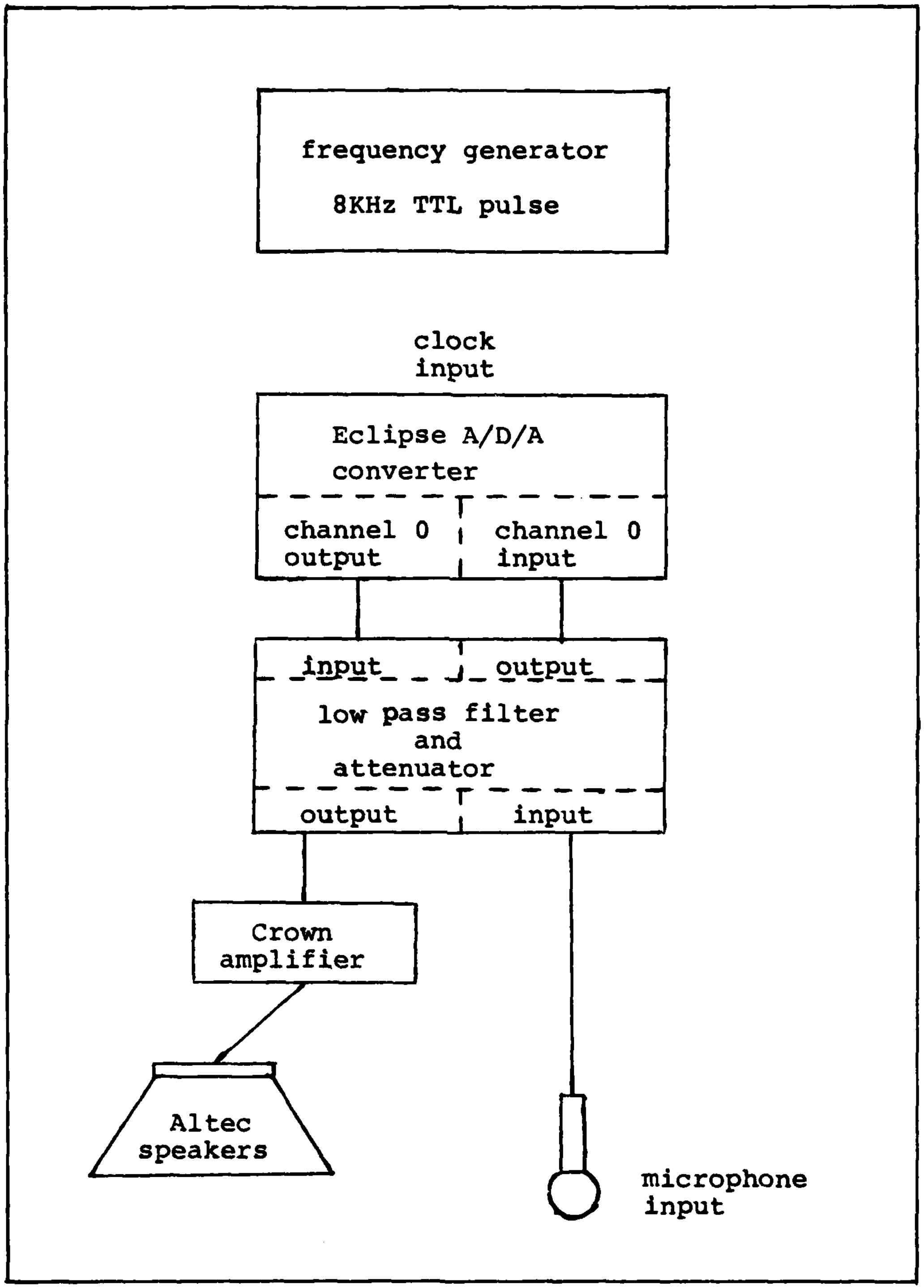



Figure 1. Eclipse A/D/A system

Source of Phoneme Template

As a baseline, Seelandt's original set of prototypes was used. All other prototypes used in this study were obtained by taking some combination of vectors from the original set. The reasoning behind this was to use the original prototype set as a basis for comparison. Because of the subjective nature Seelandt used to pick his phoneme prototypes, another set of prototypes could have been chosen using different speechfiles and/or using different time slices or vectors. It was decided at the onset that this would consume too much time and that the original set would serve adequately as a baseline.

Originally, an indirect method was used to generate the new prototype sets. Seelandt has developed a speechfile made up of just his 71 phonemes call "PHONEMES". This speechfile was used in speech synthesis (Ref 2). Running this file through TEKTALK created the frequency component file and allowed visual inspection of the speech. PHOGEN was then used for creating the phoneme templates by allowing desired vectors to be chosen from the frequency component file.

While analyzing the spectrogram of PHONEMES it was found there were slight differences between the spectral components of the speechfile and those of Seelandt's prototypes (Ref 2). It was then decided that since the original objective was to use Seelandt's prototypes as a data base, that the speechfiles used to create Seelandt's prototypes would also be used to create the new prototypes. Table I shows the speechfile used and the 5 vector phoneme picked from them. Table II lists the phoneme prototype number and their associated vector numbers from the speechfiles. The frequency component files of these speechfiles were created by SGRAM and were then used by PHOGEN.

Table I

Source Speechfiles used and

their associated phoneme templates

| Speech File | phoneme numbers |
|---|---|
| FSP1B.KS | 1 through 24 |
| FSP4A.KS | 25 through 30<br>57 through 62 |
| FSP2A.KS | 31 through 56 |
| FSP3A.KS | 63 through 71 |

Table II

Five Vector Phonemes and the vector numbers they are associated with in the original speechfiles.

| Phoneme number | Beginning Vector number | Phoneme number | Beginning Vector number |
|---|---|---|---|
| 1 | 1 | 21 | 201 |
| 2 | 9 | 22 | 211 |
| 3 | 15 | 23 | 221 |
| 4 | 27 | 24 | 230 |
| 5 | 34 | 25 | 87 |
| 6 | 41 | 26 | 96 |
| 7 | 50 | 27 | 102 |
| 8 | 56 | 28 | 108 |
| 9 | 61 | 29 | 118 |
| 10 | 67 | 30 | 125 |
| 11 | 72 | 31 | 25 |
| 12 | 104 | 32 | 32 |
| 13 | 113 | 33 | 47 |
| 14 | 118 | 34 | 61 |
| 15 | 127 | 35 | 69 |
| 16 | 134 | 36 | 81 |
| 17 | 139 | 37 | 87 |
| 18 | 148 | 38 | 96 |
| 19 | 190 | 39 | 104 |
| 20 | 195 | 40 | 113 |

Table II cont.

| Phoneme number | Beginning vector number | Phoneme number | Beginning vector number |
|---|---|---|---|
| 41 | 122 | 61 | 40 |
| 42 | 127 | 62 | 46 |
| 43 | 135 | 63 | 64 |
| 44 | 140 | 64 | 73 |
| 45 | 148 | 65 | 80 |
| 46 | 154 | 66 | 87 |
| 47 | 163 | 67 | 93 |
| 48 | 169 | 68 | 99 |
| 49 | 183 | 69 | 104 |
| 50 | 186 | 70 | 114 |
| 51 | 200 | 71 | 1 |
| 52 | 210 | | |
| 53 | 217 | | |
| 54 | 223 | | |
| 55 | 228 | | |
| 56 | 235 | | |
| 57 | 6 | | |
| 58 | 11 | | |
| 59 | 18 | | |
| 60 | 23 | | |

Single Vector Phoneme - Template Selection

The first prototype set created was the single vector phonemes. These were created by taking the middle vector from the original 5 vector phoneme template and using it as a template for that phoneme. The middle vector was chosen because in a five vector phoneme, the difference between the middle vector's frequency components and those of the vector's farthest from it are likely to be less than the first vector's frequency components compared to the fifth vector's component. Because the spectral components differ greatly from phoneme to phoneme, this is not always true. There are some phonemes that have spectral components that don't change very much as the time slice changes, an example of this is the "n" sound in "one". In this case, any one of the five could perform just as well as a phoneme template.

Two Vector Phoneme Template Selection

Initially, two sets of templates were created to compare with each other and the other template sets. The first set consisted of templates created by using the second and middle vector from the five vector templates. The second set was created from the first and second vector of the five vector templates. The comparison testing consisted of using each template set as the phoneme prototypes in PHDIST and running the recognition routines against the speechfiles the original phoneme prototypes were taken from. As expected, it showed that the majority of the phonemes performed almost identically on the given speechfiles. There were isolated cases where one phoneme of a given set performed better than the corresponding phoneme of the other set. The method used in the comparison will be covered in chapter III. With these results, a third set of phoneme templates was developed using the phoneme templates from the first two sets. If the two phonemes had identical performance, it was just an arbitrary choice as to which one was used. If one showed an improvement in performance over the other it was chosen as the phoneme template for the third set. What resulted was a hybrid set of phoneme templates that contained the best performers of the two sets of phoneme templates, and thus would have an optimized performance for two vector phoneme prototypes. Since the locations of the vectors used as prototype phoneme templates were now know, a set of phoneme templates could be constructed with relative ease and in a short period of time. With this in mind and for the sake of completeness, two more sets of two vector phoneme templates were constructed for comparison testing against the third set of phoneme templates. The fourth set was made up of phonemes containing the middle and fourth vectors and the fifth set was made up of phonemes containing the fourth and fifth vectors of each original five vector phoneme template. The same type of comparison testing was performed using these last three sets of phoneme templates. As in the first set of tests,the best performers from the three sets were combined into a sixth and final set. If there were no clearly superior performer for a given phoneme from the three sets, the template from the third set was arbitrarily chosen to be included in the sixth set. The reason for this judgement call was strictly for convenience. The program used to create the phoneme template set is constructed to either create a totally different set of templates or it can modify an existing set of templates, thus saving time and eleminating a lot of tedious work that increases the chances for an error to occur. This sixth and final set of phoneme templates is the set that was used in the comparison test between the various length phoneme templates. Table III shows a listing of phonemes in this final set, labelled set F, and the orginating template set for each phoneme. Set A is the first set, B is the second set, C is the third set, D is the fourth set and E is the fifth set.

## Table III

Final Two Vector Phoneme Templates and Template Set They Originated From

| Vector number | Originating set | Vector number | Originating set |
|---|---|---|---|
| 1 | C | 20 | A |
| 2 | E | 21 | A |
| 3 | A | 22 | E |
| 4 | D | 23 | E |
| 5 | A | 24 | E |
| 6 | D | 25 | B |
| 7 | E | 26 | D |
| 8 | E | 27 | E |
| 9 | E | 28 | A |
| 10 | E | 29 | B |
| 11 | B | 30 | B |
| 12 | A | 31 | B |
| 13 | A | 32 | B |
| 14 | A | 33 | B |
| 15 | A | 34 | B |
| 16 | A | 35 | D |
| 17 | A | 36 | A |
| 18 | A | 37 | E |
| 19 | A | 38 | A |

Table III Cont.

| Vector number | Originating set | Vector number | Originating set |
|---|---|---|---|
| 39 | A | 60 | D |
| 40 | B | 61 | A |
| 41 | B | 62 | E |
| 42 | A | 63 | A |
| 43 | A | 64 | E |
| 44 | A | 65 | A |
| 45 | D | 66 | A |
| 46 | E | 67 | D |
| 47 | E | 68 | D |
| 48 | A | 69 | A |
| 49 | A | 70 | E |
| 50 | A | 71 | C |
| 51 | B | | |
| 52 | A | | |
| 53 | A | | |
| 54 | A | | |
| 55 | A | | |
| 56 | A | | |
| 57 | E | | |
| 58 | A | | |
| 59 | A | | |

## Three Vector Phoneme Template Selection

For the comparison testing, a phoneme template set consisting of the middle three vectors from the original phonemes was created. The set was created using the same procedure used to create the one and the two vector phoneme template sets. Only one set of three vector phonemes was created. The reasoning behind creating only one set was due to the tremendous amount of overlapping that would occur in the creation of different sets of templates. What this means is that if another set of phonemes was created, only one vector per phoneme would be different from the first set of three vector phonemes. This single vector difference amounts to only 8 msec of speech. This overlapping would be expected to produce the same results in the recognition routines when compared to the set of three vector phoneme templates that were used in this project.

## Verification of Phoneme Template Sets

A verification process was initiated after the selection of each set of phoneme templates. This process was necessary to insure that the vectors used in the set of templates match the desired vectors in the original template set. The verification was done by visual inspection of the spectrograms. Because the phoneme template sets were actually record files containing the Discrete Fourier Transformed frequency components of speech, a modified spectrogram program had to be used to display the spectrogram of each phoneme template set. This program was SGRAM 4 developed by Seelandt to be used specifically for this purpose. Once the spectrograms were obtained, a vector by vector comparison was done between the phoneme template set and the original five vector phoneme template set. This verified that wrong vectors were not inadvertently chosen and that corrections made to the template sets were indeed made.

## III. Qualitative Analysis of Phoneme Template Sets

### Method

Once the different template sets were formed, a method for comparing them had to be devised. This involved the use of software tools and visual inspection. The one major problem to overcome was what to use as a model for comparison and what parameters were to be used in the comparison. Once this problem was resolved, software could be written to suit the needs dictated by the comparison tests.

As mentioned earlier, the intent of this project was to find out if five vector phoneme templates were indeed the best choice for prototypes. Based on this premise, the five vector phoneme template set was chosen as the basis for comparison. Accordingly, the speech that was used for forming the phoneme template set was also used as the standard for exercising the different phoneme template sets. This procedure was chosen because, with speech, there can be no standard universal model for comparison. It was felt that using the five vector phoneme templates in the recognition routines would produce the optimum results. These results then served as the standard model in the comparisons.

### Software Tools

With the basic ground work laid for the comparison testing, programs could be developed and utilized to provide the results needed to perform the final analysis of the comparison testing. Initially, the phoneme recognition routines used were Seelandt's TRYDIST 5 and LISTER 4 programs. TRYDIST 5 calculated the distances between the desired phoneme template set and the chosen speechfile. Each phoneme is compared to a segment of speech equal to the length of the template. The speechfile has been processed by a Descrete Fourier Transform into 8 msec vectors in the same format as the templates. In this way TRYDIST 5 then increments the segment of speech one vector at a time and performs another distance measure against the given phoneme template. In this manner, each phoneme is compared against the entire speechfile; giving a distance score at each vector excluding the last N vectors where N is the number of vectors in the phoneme prototypes. LISTER 4 then takes the results from TRYDIST 5 and outputs the five closest matches according to their distances for each vector. It also outputs a range factor and scale factor for each vector, but these were not utilized in this project (Ref 2).

Problems were encountered when these programs were first used. In order to allow the program to work with various length phoneme templates, TRYDIST 5 had some optional statements that needed to be compiled. When an attempt was made to compile this program, compile errors occurred. By troubleshooting the program, the error was found to be two illegal statements that were used only when the compile option was chosen. These referred to a specific file that was not used for any purpose in the program. With all statements referring to this file eliminated, the program compiled successfully.

After working with TRYDIST 5 and LISTER 4, another problem surfaced that lead to the elimination of these programs as a primary means of performing the recognition routines. This problem was in the output of LISTER 4 where the top five choices were printed for each vector. The output routine listed the top choices by the phoneme name (i.e. ZX, AH, IY, etc.). This proved to be unsatisfactory for this project since the same name was given to more than one phoneme template in many cases. An example of this is the "T" sound in both "two" and "eight". "Two" has three phonemes that were named "Tx". Fortunantly, Montgomery also realized this and modified TRYDIST 5 to create PHDIST and modified LISTER 4 to create CHOICE 5. These programs perform the same function as Seelandt's program, but the output would be the top phoneme template choice for each vector as a phoneme number. This made it easy to see exactly which phoneme was chosen (Ref 3).

As good as the output of CHOICE 5 was for giving an indication of the phoneme templates' performance, it still was not in a desirable form for documenting the results of the comparison. A more qualitative and visual indication of the test was needed as documentation. It was felt that the best method for meeting these parameters was using graphical representations of the results. This task was performed by the program, PLOTPHON. In this program the results from CHOICE 5 were utilized. The user chooses the phoneme to be examined and the program searches the results from CHOICE 5, looking for where this chosen phoneme shows up. If the phoneme appears more than twice it then proceeds to the next step of the program, otherwise it informs the user that the phoneme appears less than twice and ends the program. The next step involves modifying the phoneme's distance scores to lie in an amplitude value between one and zero. This is done by using the following equation:

$$A_p = \frac{N_p - Mmin}{Mmax - Mmin}$$

$A_p$ - Amplitude of distance score for vector p.

$N_p$ - Phoneme distance score for vector p.

Mmin - Minimum distance score in speechfile for given phoneme.

Mmax - Maximum distance score in speechfile for given phoneme.

Once the computation has been performed, the amplitudes are then plotted on a graph versus their corresponding vector numbers from the speechfile.

A modification of this program also allows the user to construct a window illustrating where, in a speechfile, a phoneme was taken from. This was useful in showing where a given phoneme was expected to appear when the original speechfile was used in the recognition routine. With a graph for a given phoneme number and from each set of templates, the results can now be easily compared and with a hard copy of the graph, the results are also documented. Examples of the resultant graphs are shown in Figure 2 through Figure 5. A more extensive documentation of the results is in Appendix B.

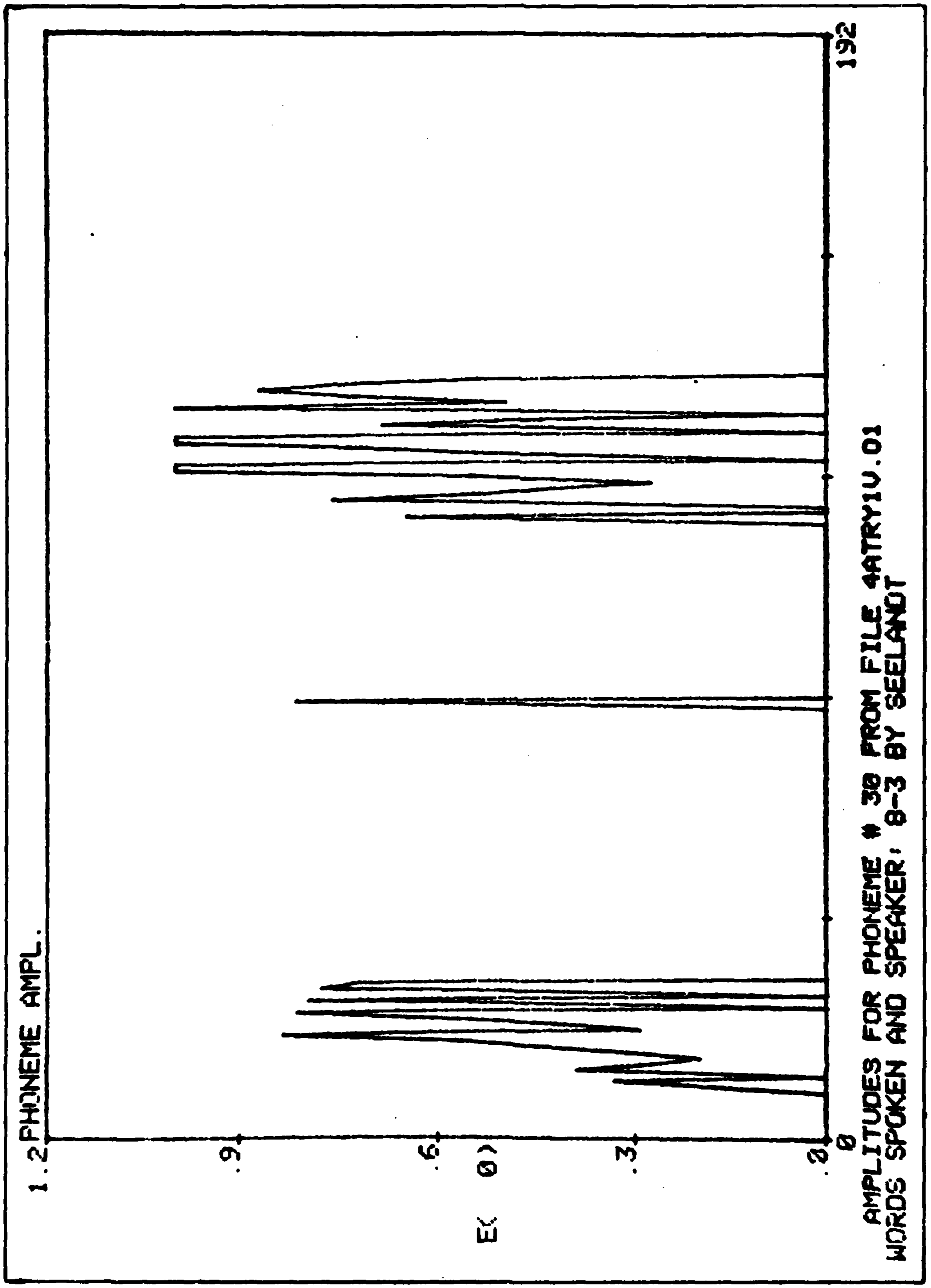



Figure 2. Recognition results using single vector phonemes

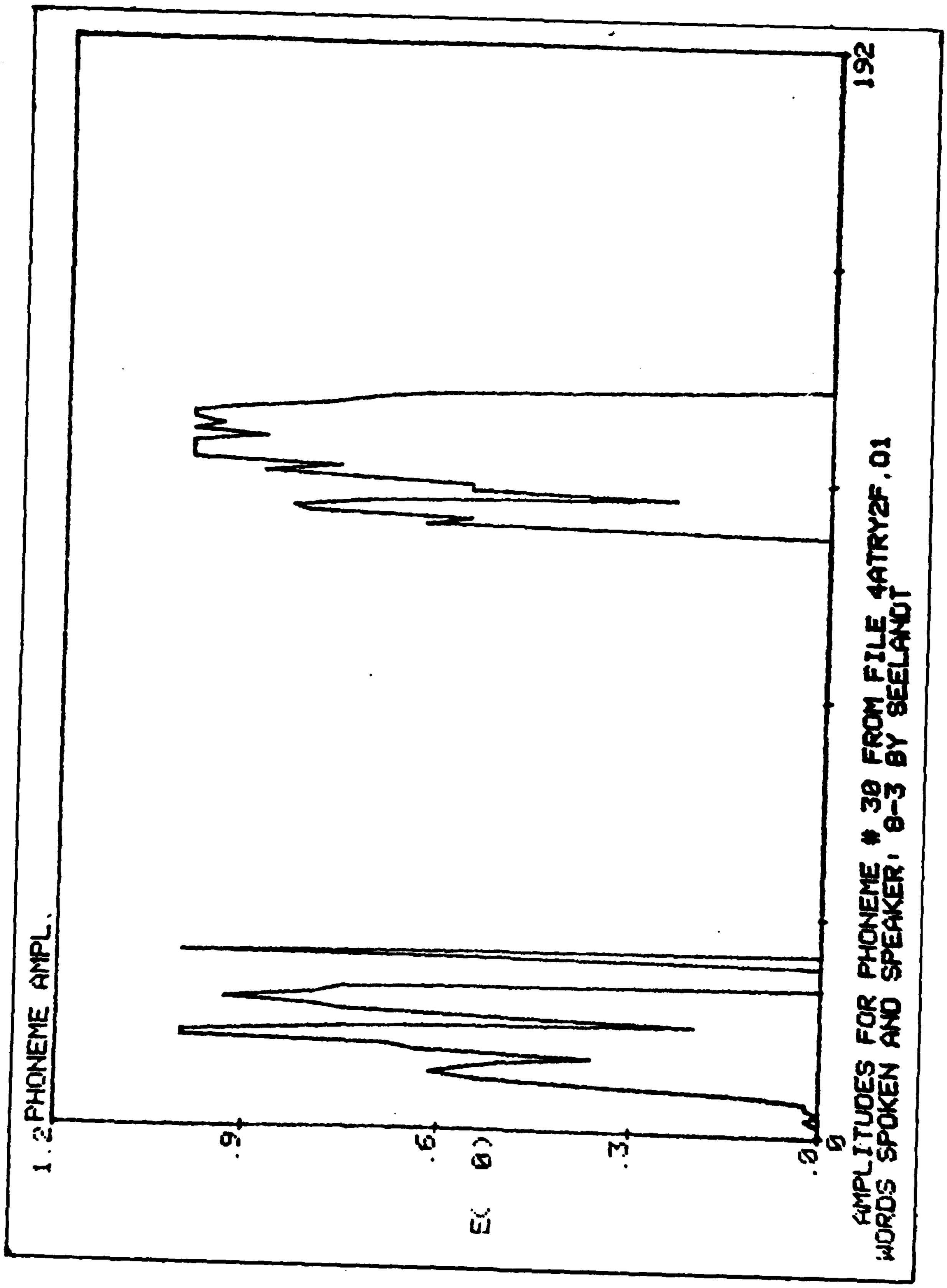



Figure 3. Recognition results using two vector phonemes

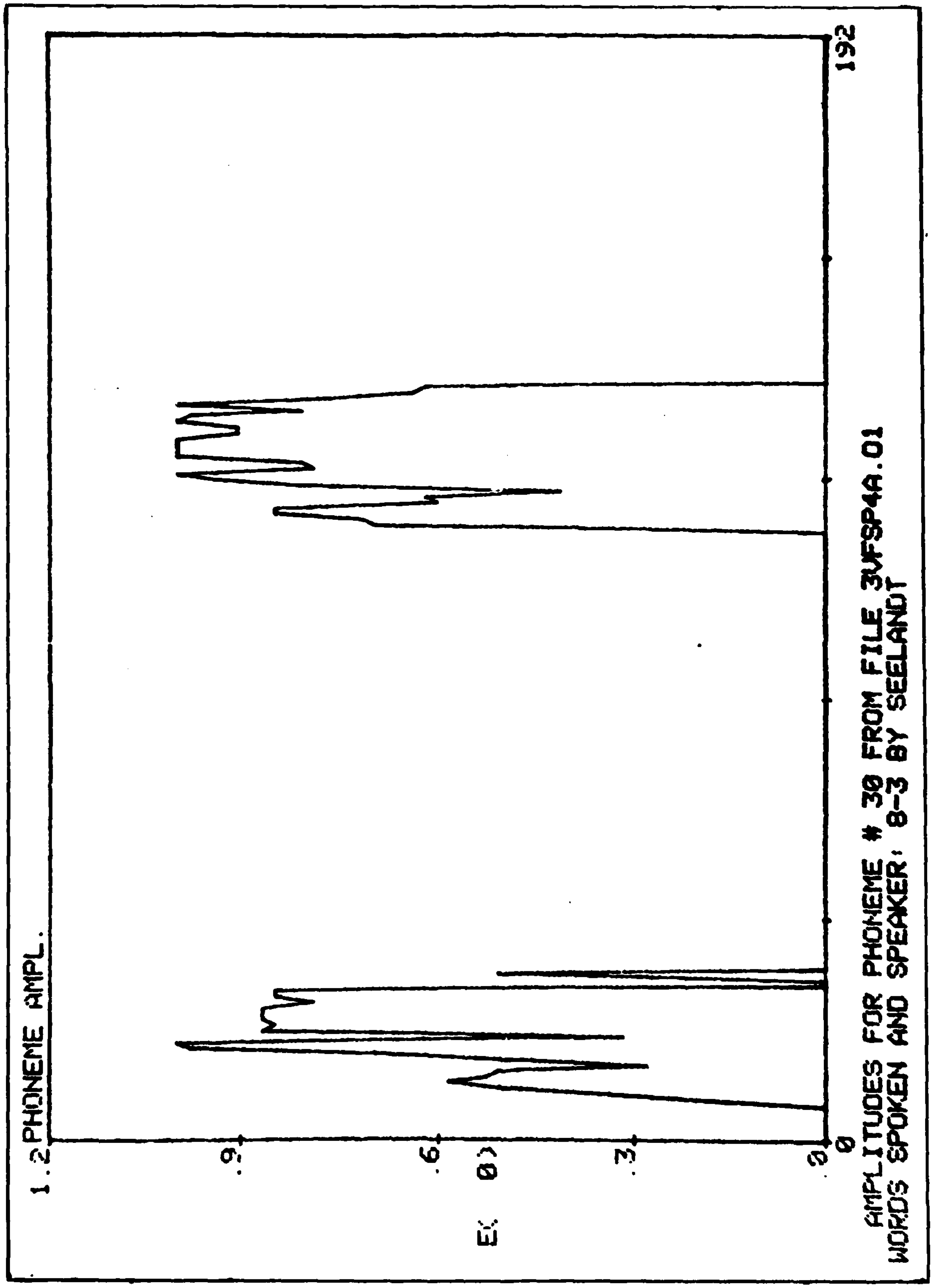



Figure 4. Recognition results using three vector phonemes

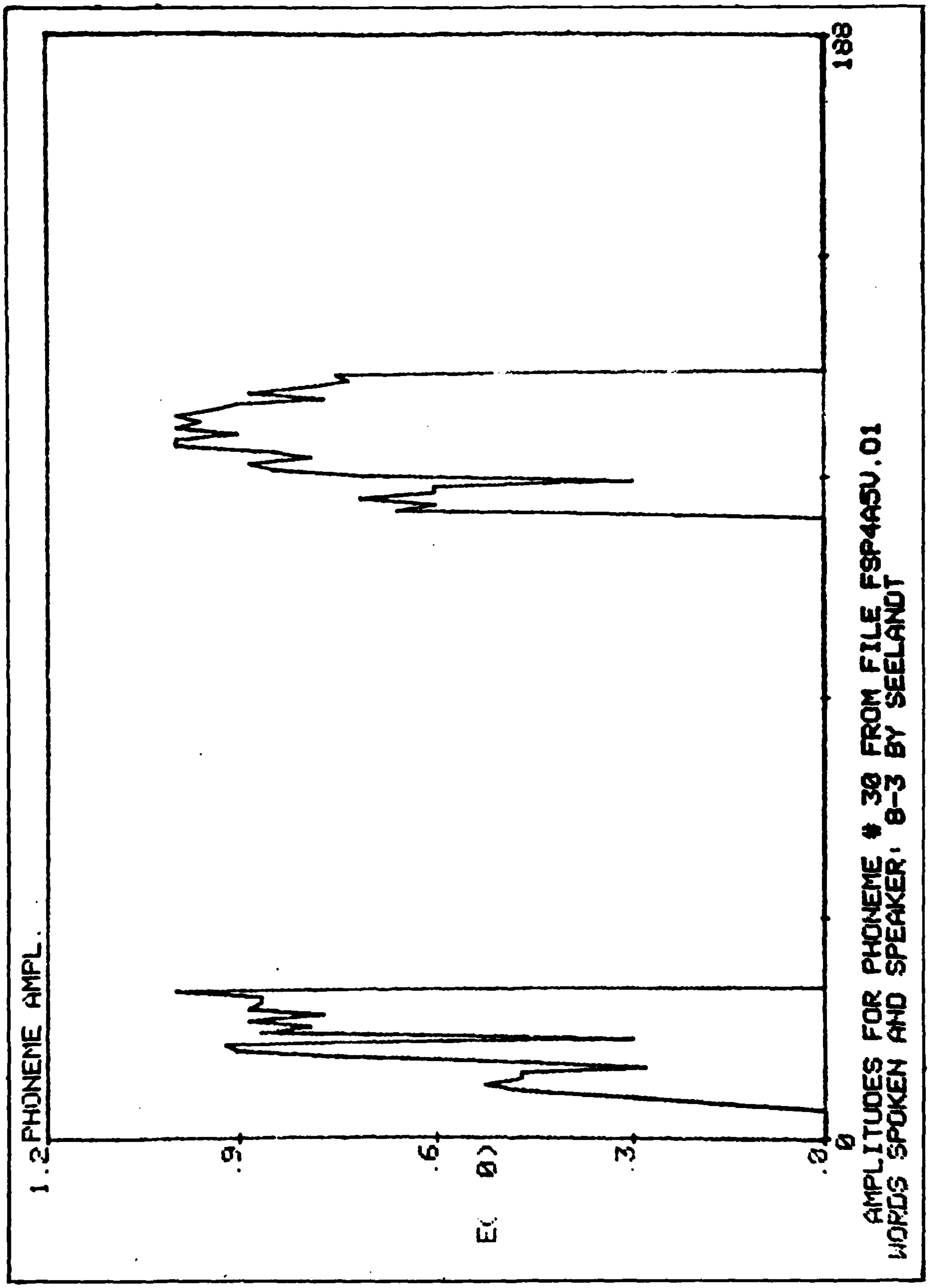



Figure 5. Recognition results using five vector phonemes

## IV. Speech Synthesis

As results from the recognition routines were being obtained, it was suggested that a way be devised to output the results as speech. This would give an audible indication of how the recognition routine performed. It also would serve as another metric for measuring the performance of the phoneme template sets. Albeit, it is a very subjective metric because it is based on what is perceived through each individual's auditory senses.

### Method

In human speech, an utterance consists of a combination of sounds from a basic set of sounds that are called phonemes. For synthesis the same principle is used. The computer has stored in memory a record file of the basic sounds needed to produce an utterance. For this project, the question was raised as to how much information is needed in each phoneme to produce understandable speech. The best way to answer this question was to start at the smallest number of vectors per phoneme and build upon this basic building block until understandable speech has been acheived. Therefore, the first file of phonemes used in synthesized speech was created as a record file of digitized speech using only the middle vector from each of the 71 five vector phonemes.

## Software Tools

The tool used to create this file was a program called SPEECH. This program allowed the user to select the desired vectors in a speechfile and put them in another file for use with synthesized speech. A spectrogram of the speechfile of single vector phonemes is illustrated in Figure 6.

After the file of phonemes was created, a coherence test was devised to compare the single vector phonemes to the five vector phonemes. Before a direct comparison could be made, the single vector phoneme set had to be modified. This was due to the fact that each phoneme represented only 8 msec of speech and if the speechfile was to be an output of speech it would be unintelligable. What was needed was a procedure to lengthen the speechfile by repeating each phoneme a suitable number of times so that the length of the modified phoneme speechfile corresponded to that of the five vector phoneme speechfile. The creation of this file was done in a two step process. The first step involved utilizing one of Seelandt's programs call TYPETALK. This program created a file of vector numbers corresponding to those found in a speechfile and a phoneme number that was repeated five times. An example of the format for this file is as follows: Vector one would be followed by phoneme one, as would vector two through five; Vector six would be followed by phoneme two,

**********"PHONEMES." USED IN TALK2.SV; SINGLE VECTOR PHONEMES

DATE: 11 29 1982
TIME: 20 45 53
NO. OF SAMPLES= 64 FIRST BLOCK= 0 LAST BLOCK= 17
HIGH FREQUENCIES PREEMPHASIZED, DB/OCTAVE= 6
PREEMPHASIS STARTS AT FREQ: 500.
ENERGY THRESHOLD= 1.0

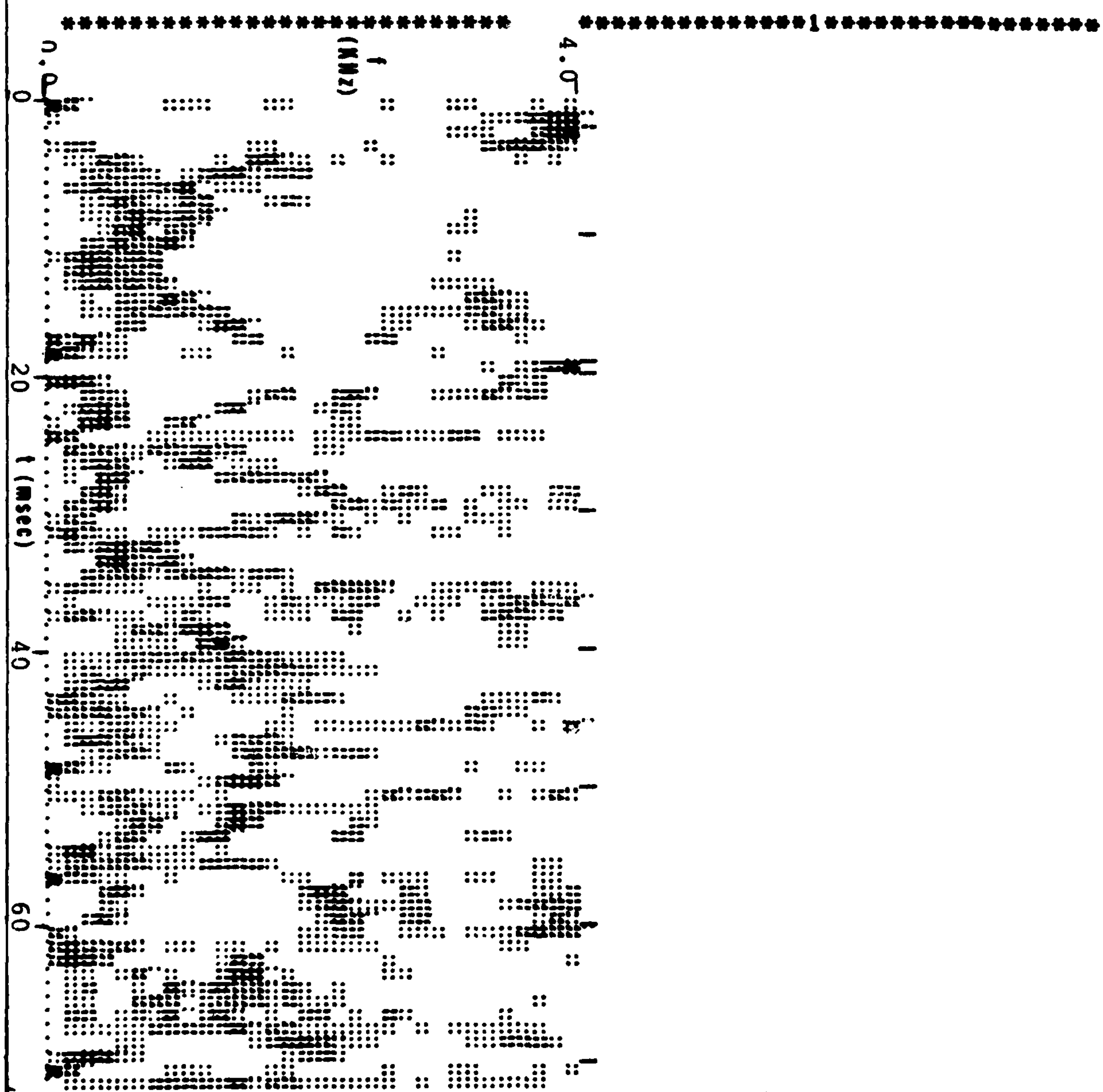



Figure 6. Single vector phoneme speechfile

as would vector seven through ten; and the format would continue through the seventy-first phoneme.

Upon creation of this file, program TALK 2 would be used to create the speechfile. This is also the program that utilizes the speechfile of single vector phonemes that was originally created by SPEECH. TALK 2 takes the file created by TYPETALK and reads each phoneme number in sequence and goes to its phoneme speechfile and selects the corresponding phoneme from that file and puts it into a user designated speechfile. The resulting end product is a speechfile consisting of 71 groups where each group consisted of a single vector phoneme repeated five times. For further clarification of this description, Figure 7 illustrates the resulting speechfile as a spectrogram. In contrast, Figure 8 illustrates the spectrogram of the speechfile made up of the 71 five vector phonemes. As seen in the figures there is a marked difference between the two speechfiles. With the five vector phonemes, the file contains enough of the speaker's characteristics for a listener to identify the speaker even though a large portion of the original speechfile was not used when the phonemes were developed. This is true because when Seelandt picked his phonemes, he was looking for changes during an utterance. Therefore, the phoneme speechfile sounds slightly choppy and the individual sound (i.e. phoneme) is not held for the same length of time as in the uttered speech.

When the modified single vector phoneme speechfile was listened to it was coherent speech, but had a very mechanical sound and was also choppy. There was none of the recognizable speaker's characteristics as with the five vector phonemes. This was expected because this speechfile contained only one-fifth of the information contained in the five vector phonemes. However, there was still enough information to identify what was said after each single vector phoneme was repeated five consecutive times.

The results using the modified single vector phoneme speechfile were encouraging enough to use the single vector phonemes as the source for synthesized speech. This precluded testing the two or three vector phonemes and made the algorithm for synthesizing speech easier to deal with. The program that employed the synthesized speech algorithm was TALK 2. This algorithm was explained earlier, but in this case TALK 2 reads a file containing the vector numbers from a speechfile and their corresponding phonemes chosen in the recognition routines. In this manner, the results from the recognition routines becomes an audible output that is fed back tc the user so he or she can make a comparison between results obtained using different phoneme templates and make the judgement as to which sounds the best. As mentioned earlier, this method is some what subjective, but did provide another means to compare the four sets of templates.


Figure 7a. Synthesized five vector phonemes

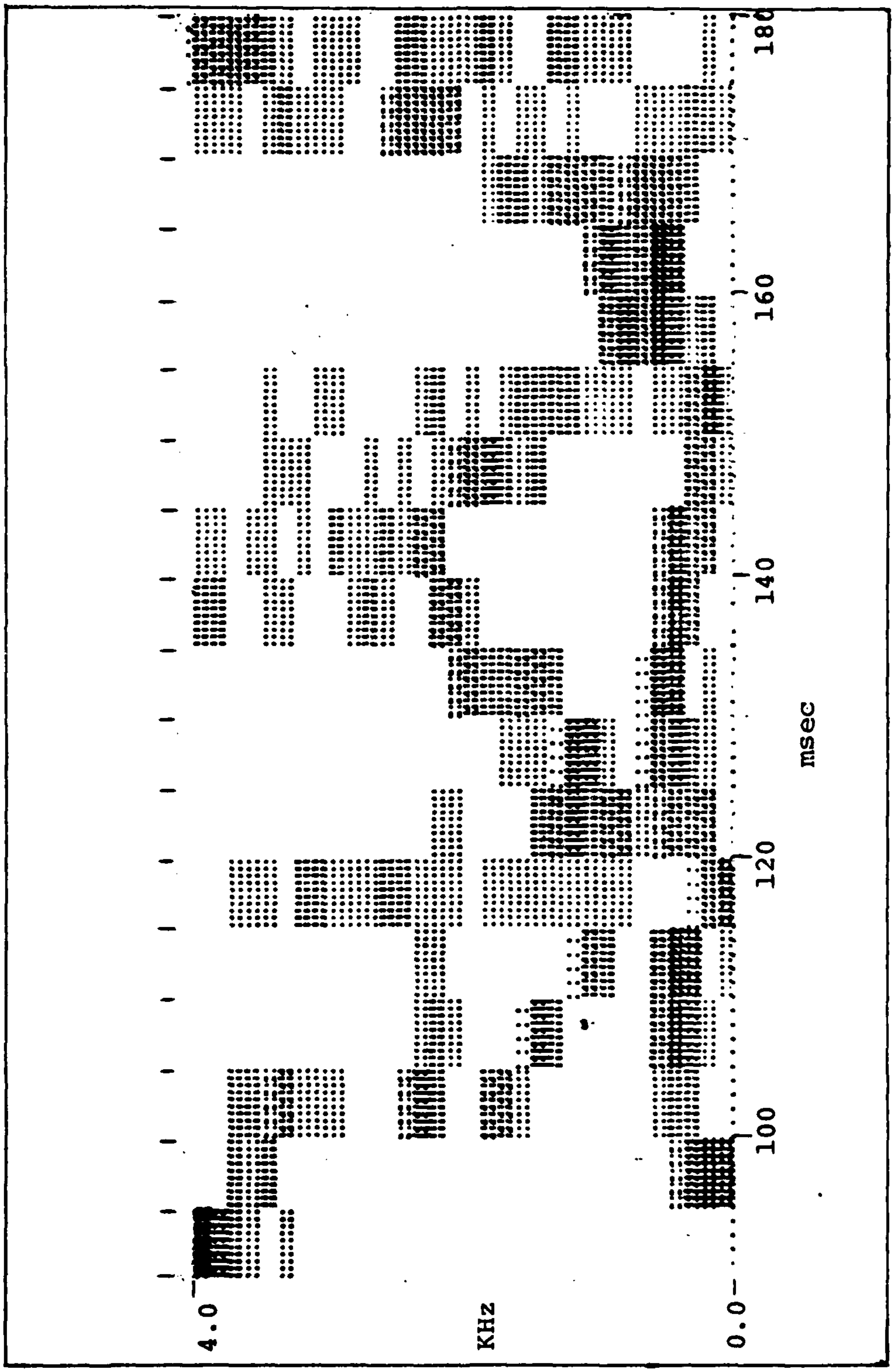



Figure 7b. Synthesized five vector phonemes

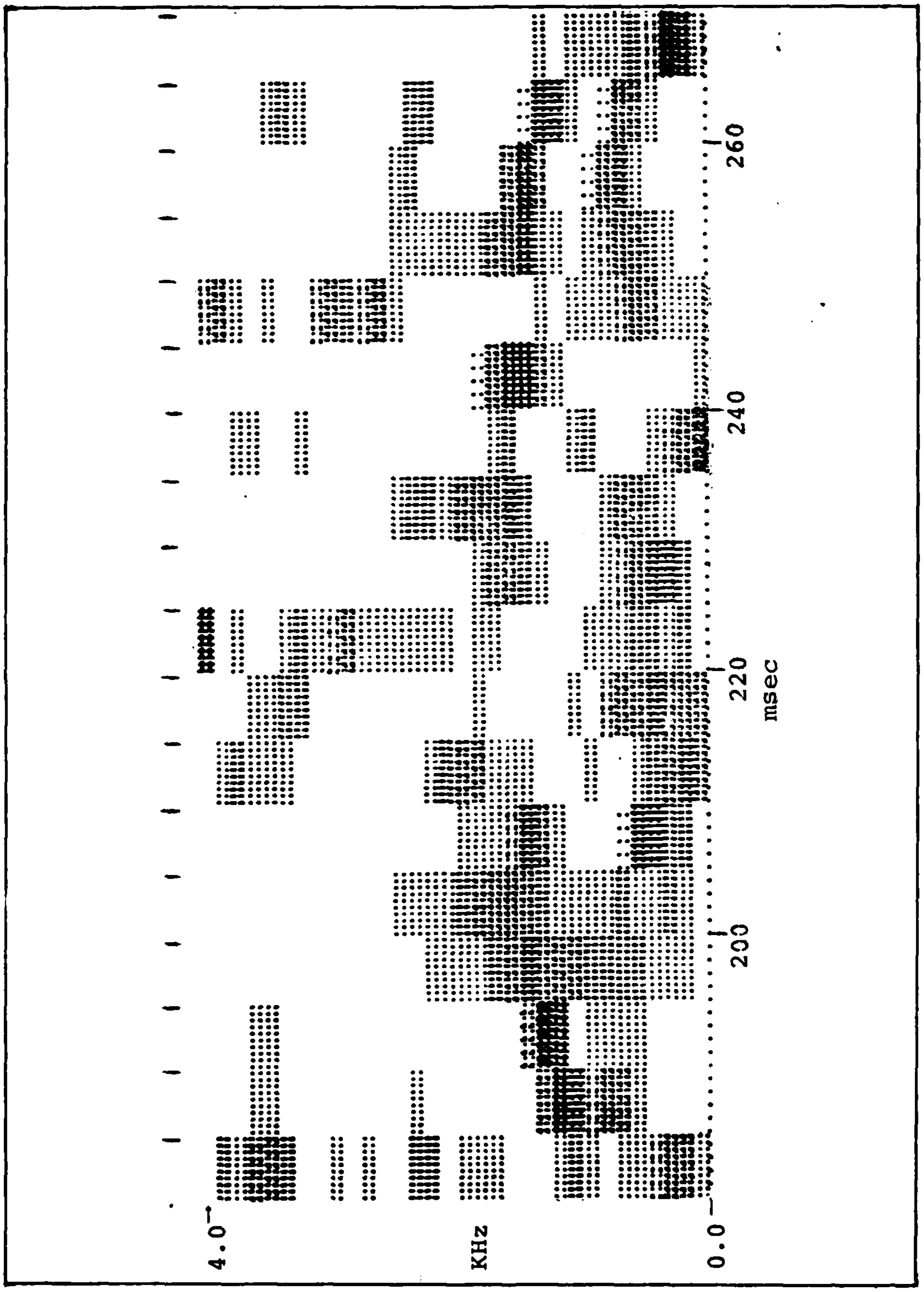



Figure 7c. Synthesized five vector phonemes

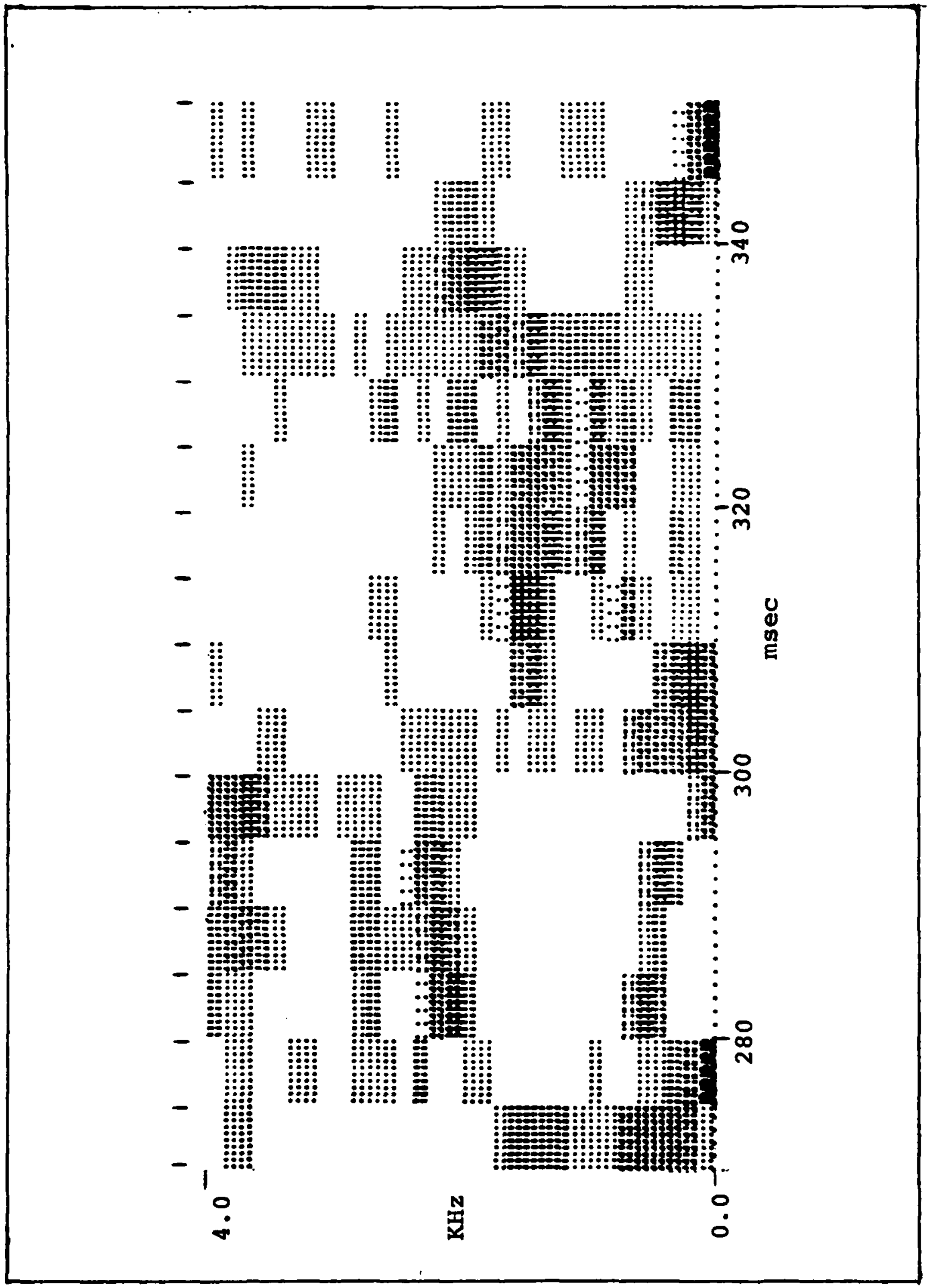



Figure 7d. Synthesized five vector phonemes

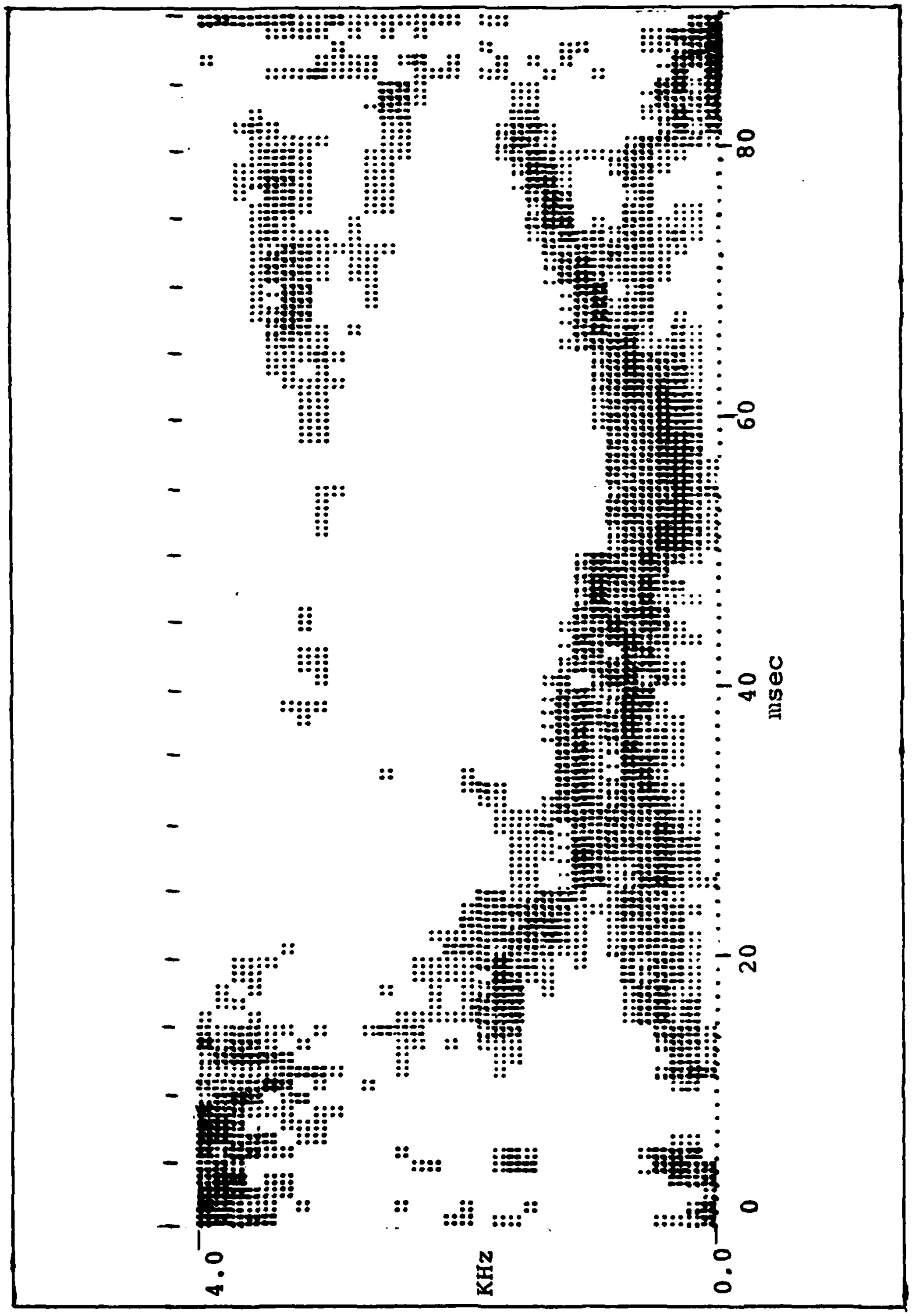



Figure 8a. Five vector phonemes


Figure 8b. Five vector phonemes

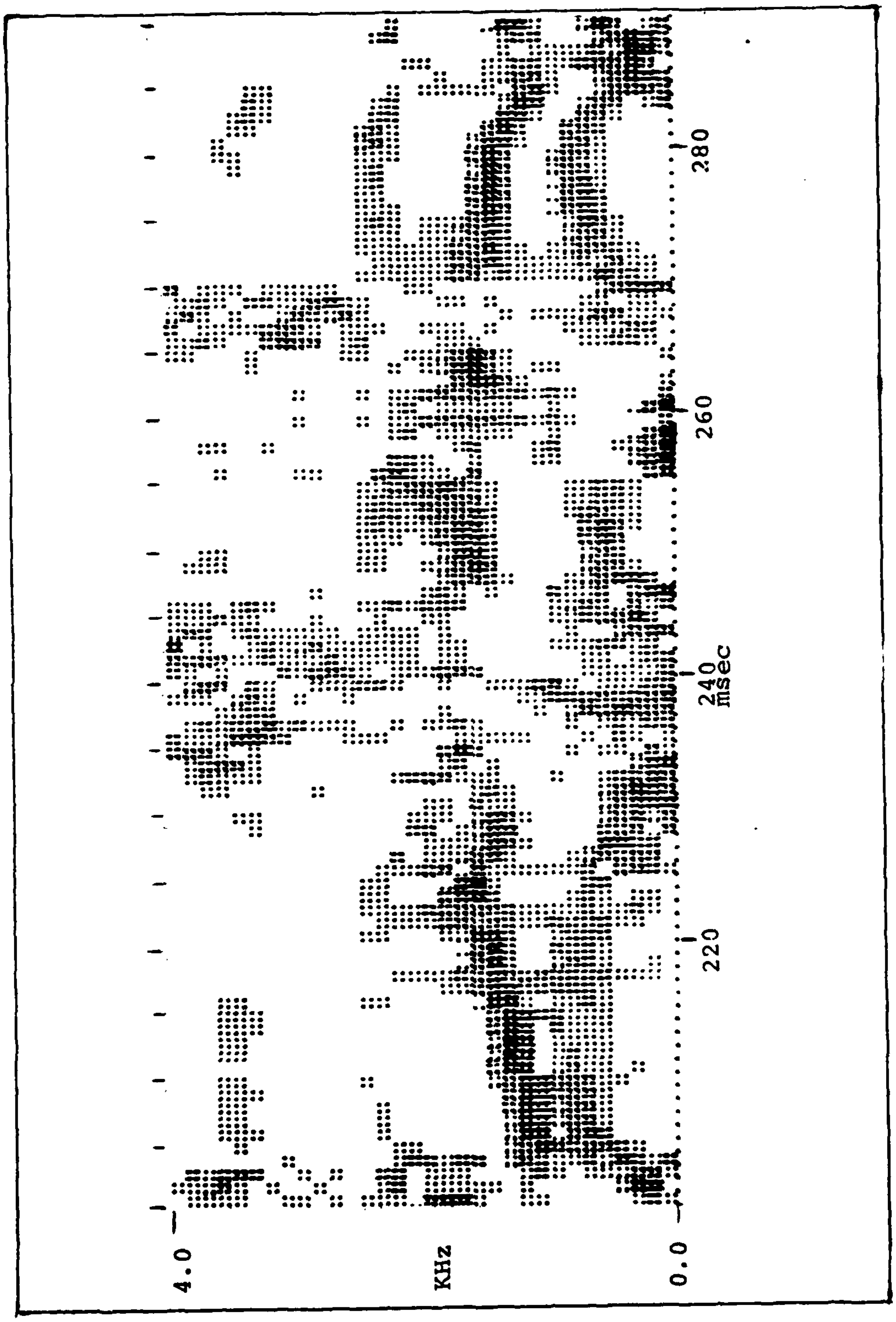



Figure 8c. Five vector phonemes

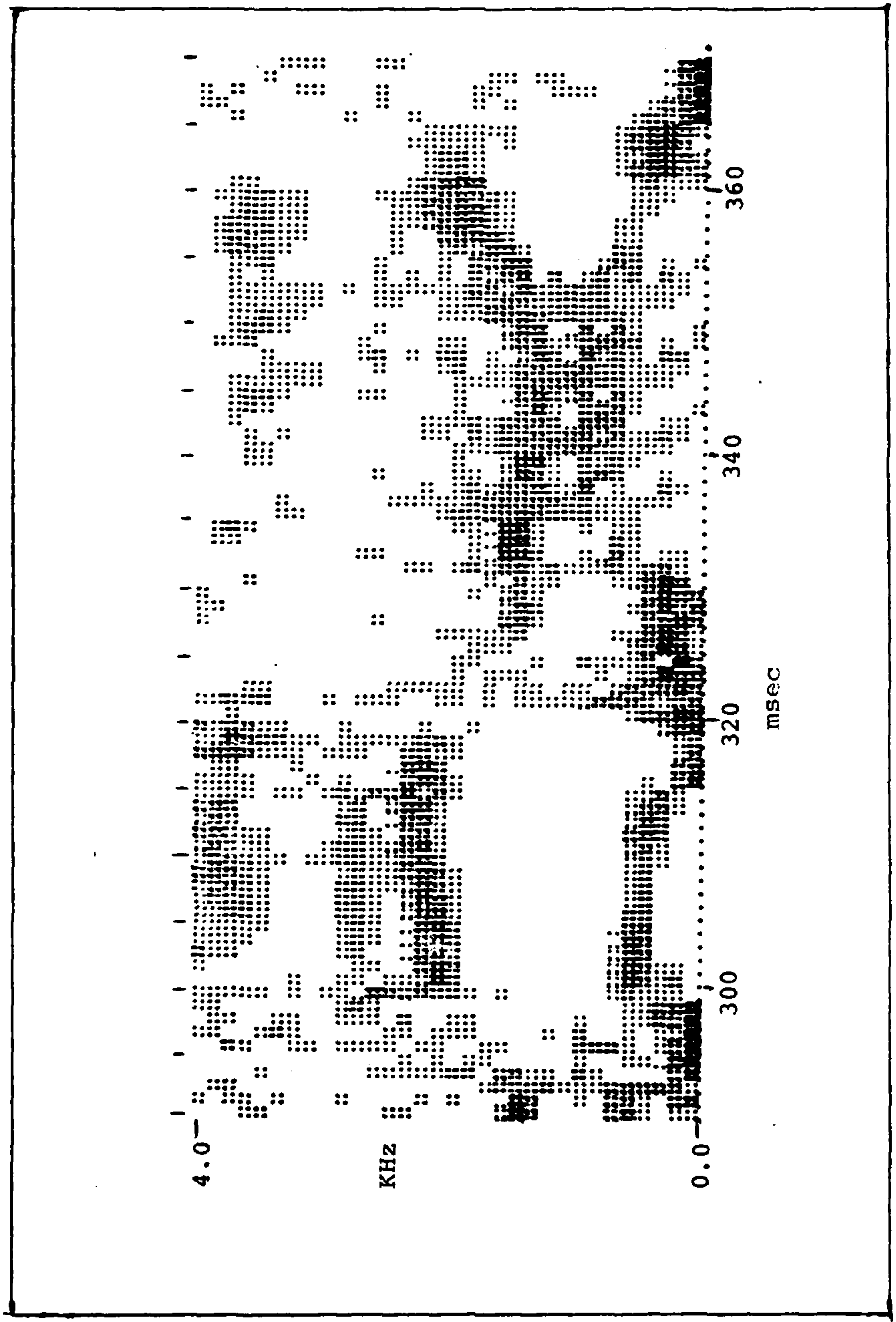



Figure 8d. Five vector phonemes

## V. Implementation of Martin's Recognition Routine (Ref 6)

In Martin's thesis, there is developed a recognition routine that was used with speech that was processed using the array processor (Ref 6). There are three major differences between Martin's program and Seelandt's. The first difference is that Martin's program could only use a single phoneme template set in the recognition process. The second difference is that Martin's program could use the speechfile as a template set and use the speechfile in the recognition routine and run it against itself. This is very useful in investigating similarities in the speechfile where a vector may be a close match to another unrelated vector in the utterance. The final difference involved the method by which distances were computed between the template and the associated block of speech. Both used the general Minkowski metric of order s, where:

$$\partial_m\left( \xi_k \quad \xi_l \right) = \left[ \sum_{j=1}^{d} \left| \xi_{kj} - \xi_{lj} \right|^s \right]^{1/s} \qquad \text{(Ref 8)}$$

is the general formula. Seelandt used the Minkowski distance with the order equal to one. This is also known as the City Block distance or M1 distance and, in his phoneme distance. However, there were a few instances where the top phoneme choice was different. For the other four choices there were also a few instances where the choices were reversed and also where a different choice altogether showed up. In all, the differences were isolated. Both programs were checked for errors, but none were found. It was then concluded that the isolated differences were the result of round off errors in the routine. This is very likely since both programs convert integers to real numbers and real numbers to integers in various portions of the routines. When it was shown that both programs generally gave the same results, Martin's program was chosen as the primary software tool for speech recognition. This was due to the versatility of the parameters; Martin's made extensive use of menues to effect changes in the parameters. Another advantage Martin's program had was that it could process a full 88 blocks or 352 vectors of speech. This is compared to Seelandt's program which could only process 200 vectors of speech at a time. This characteristic was valuable when the results from the program were an output of synthesized speech. Seelandt's results had to be synthesized as two speechfiles which added to the distortion of the speech because of the cut off in the middle of an utterance. Because of the restriction Martin's program placed on the template set, modifications had to be made program, was of the following form:

$$D_t = 1/N \sum_{j=1}^{N} \left| s_j - s_{j-t} \right| \qquad \text{for} \quad t = 0,1,\cdots,t_{max}$$

Sj = the jth speech sample

t = time shift

N = Number of speech samples (Ref 2:53,54)

In Martin's program, the user had the choice of using the Minkowski distance with the order equal to one or two. With the order equal to two, the formula is also known as the Eulidean distance or M2 distance. The equation for the formula is as follows:

$$\partial_E(\xi_k, \xi_l) = \left[ \sum_{i-1}^{d} (\xi_{ki} - \xi_{li})^2 \right]^{1/2}$$

$$= \left[ (\xi_k - \xi_l)^t (\xi_k - \xi_l) \right]^{1/2} \quad \text{(Ref 8:231-240)}$$

Using both programs and the single vector phoneme template set, both recognition routines were run using the same speechfile. The top five phoneme choices per vector were then compared to note any discrepancy when the City Block distance was used in both programs. Ideally, both outputs should be identical since all parameters in both programs were also identical. Generally the outputs did track each other in choosing the top to accommodate the use of different length phoneme templates. Two options were available; one was to modify the program and the other was to modify the files used by the program. Because of time constraints and possible interface problems it was decided to alter the speech and phoneme templates' spectrum files.

In order to explain how the alterations were accomplished a review of how the recognition algorithm operates for phonemes greater than one vector and how Martin's program utilizes the file created for the templates and speech is needed. As stated in Chapter III, the recognition algorithm operates by comparing a phoneme to the speechfile in a fashion similar to a sliding window. The window is N vectors wide, where N is the number of vectors in the phoneme template. The window slides through the speechfile one vector at a time and has a distance value calculated for each increment. What this in effect does is lengthen the speechfile by a factor of N. All speechfiles used by Martin's program must have a spectrum of the speechfile calculated. A header block of 256 works is also added to this file. This header holds information such as speechfile name, number of components per vector, and first and last vector of the block of speech to be used in Martin's program. Martin's program inserts this information from menu selections and later uses this information to perform the recognition routines.

Alterations to the templates' spectrum file was limited to changing values in the header block. For an N vector phoneme template, the number of frequency components per vector was increased by a factor of N. Therefore if there are 32 frequency components per vector and the phoneme length was four vectors, the altered vector would have 128 frequency components. With each vector made up of four 32 component vectors, the value for the number of vectors in the template set also had to be changed. Therefore the original value in the header block was divided by N and this new value was substituted back into the header.

Alteration of the speech spectrum file was more extensive than that involving the template spectrum file. The value for the components per vector was changed to match that of the template file. The value for the number of vectors in the speechfile was also changed. In this case, a value that is one less than the number of vectors per phoneme is subtracted from the number of vectors in the unaltered speechfile. The last alteration involved the speechfile itself where it had to be altered to conform to the recognition algorithm for multiple vector phonemes. As stated earlier, the window was incremented one vector at a time. To duplicate this effect in Martin's program, the speechfile had to be lengthened by treating the vectors in the window as a single vector. In this manner, the contents in the window as it increments through the file is sequently stored in another file as the altered speechfile. An example of this procedure is shown in figure 9 for a four vector phoneme template.

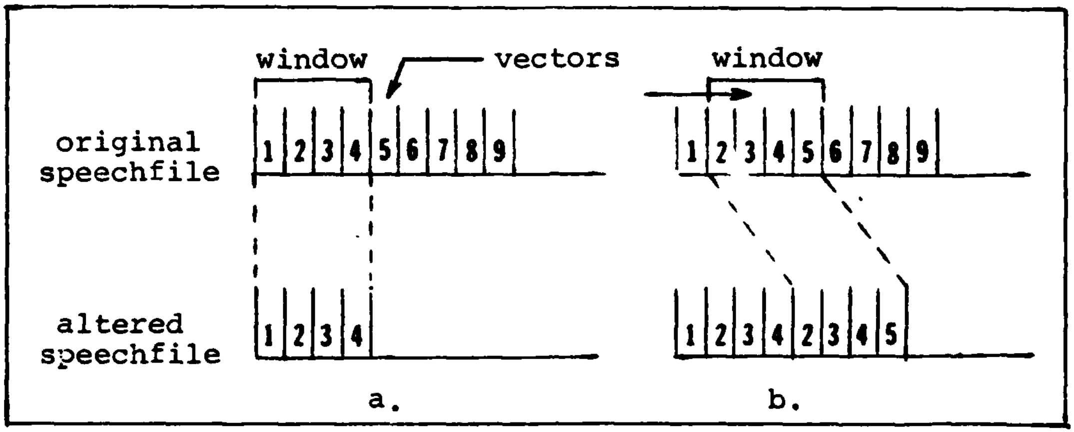



Figure 9. Four vector phoneme template

The alteration routine was performed by a program called CHNGFILE. As can be seen, if the speechfile involved was originally 88 blocks long the altered speechfile would require an enormous amount of file space. Because of the size of the file it also takes an extraordinary amount of computer time to perform the computations in the recognition routine. For an 88 block unaltered speechfile this time can be on the order of 15 to 20 minutes, even though using this procedure for multiple vector phonemes precluded most interface problems with Martin's program, one problem did surface. This problem came about when altered template sets and speechfiles were used for three and five vector phonemes. In these cases, Martin's program would run without displaying error conditions but the output would show glaring mistakes. These mistakes typically would show up as a sequence of phoneme choices that was repeated throughout the speechfile or as a single top phoneme choice that would appear at regular intervals in the output. Because of time constraints, the problem could not be located; but because it didn't show up using single, two or four vector phonemes it was concluded that the problem was tied to the number of frequency components per vector.

Since only multiples of 32 could be used for the number of frequency components per vector, a set of four vector phoneme templates was created. This set was used instead of the five vector phonemes in comparison testing using Martin's programs. Because of the difference of only one vector, it was felt that the results would still be valid in the test cases. This assumption was proven out in a comparison of results using four vector and five vector phoneme templates in Seelandt's program. With this assumption proven, the results from Martin's program were modified by Beachy's program to present a more descriptive output (Ref 7). This consisted of a listing showing the top five choices for each vector in a speechfile in a format similar to Seelandt's (Ref 2). The output from Beachy's program was then modified to create a file of the results that were compatable with TALK 2. The results from the four vector phoneme templates, were used as the basis of the comparison and established the baseline.

## VI. Results and Recommendations

Initially, the purpose of the research project was to analyze the results of speech recognition using various length phoneme templates and determine which phoneme template set gave the best overall performance. A secondary effort during the research project was the synthesis of speech using single vector phonemes as the source. The synthesized speech was also used as a performance indicator for the various length phonemes. In conjunction with the development of synthesized speech, software was developed to utilize Martin's program DRVR in the evaluation of single, two and four vector phoneme templates.

### Results

Results from the recognition routines showed that there was a gradual degradation as the length of the phoneme decreases. Table IV lists the phonemes for each template set that showed a degradation in performance when compared to the five vector phoneme template set. There was a definite pattern in the phonemes that tended to be dependent on the phoneme length. These phonemes tended to be those that had a larger variation in spectral components from vector to vector, such as with the fricative and sibilence sounds. If the phonemes had small variaties in the components from vector to vector there were no significant changes in performance between single vector or five vector phonemes. With the degradation resulting from the phonemes in Table IV, the five vector phoneme templates had the best overall performance.

The speech synthesis portion of this project almost exclusively used the output generated by Martin's program. The only time the output from Seelandt's program was used was in verifying the software associated with Martin's program. This was done by both visually checking the output and then comparing the synthesized speech by recording the two speechfiles simultaneously on the Ampex reel-to-reel recorder. The tape was then played back and by listening to the recording, a comparison could be made. This same method was applied when comparing the synthesized output for results using both M1 and M2 distances and the single, two and four vector phoneme templates. This gave a total of six separate synthesized speechfiles for each speechfile that was run through the recognition routine. Using the same speechfiles that were used for the first portion of the research, the results from the four vector templates sounded better than the other two template sets. On the other hand there was very little difference in the quality when comparing the two distances using the same template set. This was further verified by examining the printed outputs from the recognition routine. If a choice had to be made, M1 had a marginal advantage over the M2

Table IV

Phonemes from each template set that showed a degradation.

| Template set | Phoneme number |
|---|---|
| 1 vector phonemes | 2,3,9,11,20,21,25,29, 31,32,37,36,45,46,48,54 |
| also confused phoneme 65 with 66 and 3 with 2 | |
| 2 vector phonemes | 3,24,26,31,32,44,48,50, 51,67,69 |
| also confused phoneme 36 with 37 | |
| 3 vector phonemes | 20,25,31,32,37,40 |
| also confused phonemes 9 with 10, 65 with 66 and 57 with 58 | |

distance but not enough to make a significient difference in the results. The recognition and synthesis routines were then run using speech other than Seelandt's. Using a limited population of test subjects, the results were divided as to which template set produced the best results. Those that were familiar with the project picked the four vector templates as the best, but those having no experience with speech recognition tended to pick the single vector templates as the best. During the interviews the inexperienced listeners said they picked the single vector templates because there was less spurious noise in the speechfile. The experienced listeners tended to pick the four vector templates because of the better results involving the fricative and sibilance sounds. Because of this split, it also showed that as speech other than that used to pick the phonemes was used, it became increasingly more difficult to make a consistant judgement on which phoneme template performed the best.

Recommendations

In a research project such as this that was so time consuming, constraints had to be made to keep it at a more managable size. Therefore, the following recommendations are made to gain more insight into the problem of speech synthesis and speech recognition.

The first recommendation is to analyze the utterance of useful words such as frequency, step, CCIP and developing a more complete phoneme template incorporating phonemes derived from these words. This will result in a phoneme set that would be more universal and will be more useful from a practical point of view.

A second recommendation involves decreasing the processing time of Martin's program using multiple length phonemes. A routine needs to be devised that will take the output of any given phoneme template set that has been used in Martin's program and perform a summation of the distance using the equation:

$$D_n(i\ j) = \sum_{h}^{k-1} I_{i+h,\,j+h}$$

Where: n = phoneme number

k = number of vectors per phonemes

h = 0,1,2,....

i = vector of phoneme

j = vector of speechfile

I = distance value in results file from Martin's program

This will preclude altering either the phoneme template file or the speechfile. It should also decrease the program run time and save file space.

The third recommendation involves the synthesized speechfile. An attempt should be made to smooth out the spectrum of the file to avoid sharp discrepancies in adjacent vectors. In actual speech it is impossible, due to the structure of the vocal cords, to create sharp changes in sounds. There is a transition between one phoneme to the next and this factor should be simulated in synthesized speech. This can be done by incorporating a decision routine in the process that examines groups of vectors for drastic changes in spectral components.

# Bibliography


1. Potter, R. K., George Kopp, and Harriet Green. Visible Speech. New York; D. Van Nostrand Company, Inc., 1947.

2. Seelandt, Karl G. Computer Analysis and Recognition of Phoneme Sounds in Connected Speech. MS Thesis GE/EE/81D-53. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1981.

3. Montgomery, Gerard J. Isolated Word Recognition Using Fuzzy Set Theory. MS Thesis GE/EE/82D-74. School of Engineering, Air Force Institute of Technolgy, Wright-Patterson AFB, Ohio, December 1982.

4. Felkey, Mark A. Automatic Recognition of Phonemes in Continuous Speech. MS Thesis GE/EE/80D-20. School of Engineering, Air Force Institute of Technolgy, Wright-Patterson AFB, Ohio, December 1980.

5. Allen, Gordon R. Expansion of the Nova/Eclipse Digital Signal Processing System. MS Thesis GE/EE/82D-16. School of Engineering, Air Force Institute of Technology, Wright Patterson AFB, Ohio, December 1982.

6. Martin, Daviel L. Parameter Analysis of the Speech Sound. MS Thesis GE/EE/82D-46. School of Engineering Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December, 1982

7. Beachy, Keith A. Computer Recognition of Phonemes in the Presence of Cockpit Induced Stress and Noise. MS Thesis GE/EE/82D-20. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1982.

8. Devijver, Pierre A. and J. Kilther. Pattern Recognition: A Statistical Approach. London; Prentice-Hall International, Inc., 1982.

9. Diller, Timothy C. and John F. Siebenand. "Speaker Independent Work Recognition Using Sex-Dependent Clustering", IEEE 1981 Acoustiss, Speech and Signal Processing I: 180-183, 1981.

# APPENDIX A

## COMPUTER PROGRAMS

## PROGRAM SPEECH

| | |
|---|---|
| File: | SPEECH |
| Language: | Fortran 5 |
| Date: | October 19, 1982 |
| Author: | J. Fletcher |
| Subject: | Phoneme Speech file creation |
| Calling Sequence: | SPEECH |

| ARGUMENT | TYPE | PURPOSE |
|---|---|---|
| ISPEECH | String | file name of speech file |
| ILENGTH | Integer | vector length of phonemes |
| IREC | Integer | component count for each one |
| ISIZE | Integer | record size for phoneme file |
| IPHON | Integer | phoneme number selected |
| IVECT | Integer | starting vector for selected phoneme |
| INDEX | Integer | checks for best vector in a block |
| IBLK | Integer | indexes to proper block of speech |
| ISPCH | Integer Array | holds 2 blocks of speech read from a file |
| ISTART | Integer | indexes to proper vector in a block of speech |
| IARRAY | Integer Array | holds the desired phoneme's vectors pulled from ISPCH |

PURPOSE:

This program is used to select the vectors or time slices from a speechfile for a user chosen phoneme. The user can select the phoneme he wants and the number of vectors for each phoneme. The file that is created by this program, PHONFILE, can then be used as a phoneme prototype in program DRVR.

DESCRIPTION:

LOCATION: FLETCHER.DR

ARGUMENT STRUCTURE:

| | | |
|---|---|---|
| ISPEECH | = | file name, 13 characters or less in Hollerith string format |
| ILENGTH | = | no restriction as long as no end of file error is returned |
| IPHON | = | any number, 0 will end the program; there is no limit on how many can be created |
| IVECT | = | must be within the bounds of the speechfile and must be nonzero |

PROGRAM USE:

This program can use any speechfile that has been created in block format. This program will select the desired time slices from the speechfile and put them in another speechfile called PHONFILE. The record number in PHONFILE corresponds to the chosen phoneme number. This file can then be run through DRVR to create a spectrum file that is used as a prototype file in the distance measurement subroutines of program DRVR.

FILES CREATED:

PHONFILE: digitized speechfile created in a record format. Each record number corresponds to a phoneme number; in this case, there are 71 records.

```
C       ****************************************
C       ****************************************
C       **                                    **
C       **      PROGRAM:   S P E E C H        **
C       **                                    **
C       **      WRITTEN BY:                   **
C       **                                    **
C       **              CAPT JAMES E. FLETCHER **
C       **                                    **
C       **                                    **
C       **      DATE:  19 OCT 82              **
C       **                                    **
C       ****************************************
C       ****************************************
C
C       NOTE:  This program is for Fortran 5.
C
C
C
C         This program creates a speech file of phonemes from
C       existing digitized speech files that are in block
C       format.  This is done by allowing the user to extract
C       up to 5 consectutive vectors (time slices) of speech
C       from the identified speech file.  The vectors are then
C       stored in another speech file, PHONFILE, as a record
C       corresponding to the phoneme number selected by the user.
C       This program also allows the user to select a different
C       speech file by typing "0" as a phoneme number and then
C       following the program's instructions.
C
C
        DIMENSION ISPEECH(13), ISPCH(512), IARRAY(320)
        TYPE"THIS PROGRAM CREATES A SPEECH FILE OF PHONEMES,"
        TYPE"USING SELLECTED VECTORS FROM EXISTING SPEECH FILES."
        TYPE"  "
C
C
  5     ACCEPT"NAME OF SPEECH FILE TO BE USED: "
        READ(11,10) ISPEECH(1)
  10    FORMAT(S13)
        CALL OPEN(1,ISPEECH,1,IER1)     ;Channel to speech file
        CALL CHECK(IER1)
C
C
        ACCEPT"LENGTH OF PHONEMES: ",ILENGTH
        IREC = ILENGTH * 64      ;Number of components to be
                                 ; extracted from speech file.
        ISIZE = IREC * 2         ;Record size for new speech file
        OPEN 2,"PHONFILE",LEN=ISIZE     ;Open new speech file
  15    ACCEPT"ENTER PHONEME NUMBER (0=STOP): ",IPHON
        IF(IPHON .EQ. 0) GO TO 25       ;     Check for user
                                        ; wanting to stop
        ACCEPT"STARTING VECTOR NUMBER:",IVECT;  Beginning vector
                                ; of speech to be extracted
        INDEX = MOD(IVECT,4)            ;Check for vector being
                                        ; last vector in block
```

```
      IBLK = ((IVECT + 4) / 4) - 1    ;Indexing for proper
                                      ;block
      IF(INDEX .EQ. 0) IBLK=(IVECT/4)-1  ;If last vector of,
                               ;block must have different
                               ;indexing equation.
      CALL RDBLK(1,IBLK,ISPCH,2,IER2)
      IF(IER2 .NE. 1) TYPE"RDBLK ERROR = ",IER2
      ISTART = (MOD(IVECT,4) - 1) * 64  ;Indexing to get proper
                                      ; vectors, using modulo 4
      IF(INDEX .EQ. 0) ISTART=3*64   ;If last vector of block,
                       ;must have different indexing equation.
      DO 20 I=1,IREC
20       IARRAY(I) = ISPCH(I+ISTART)  ;Put desired fragment of
                                      ; speech in an array
      CALL WRITRW(2,IPHON,IARRAY,1,IER3)      ;Put in PHONFILE
      IF(IER3 .NE. 1) TYPE"WRITRW ERROR = ",IER3
      GO TO 15                 ;Return to get another fragment
                               ;of speech
25    CALL RESET
      ACCEPT"DO YOU WANT ANOTHER SPEECH FILE?(1=Y/0=N) ",IANS
      IF(IANS .EQ. 1) GO TO 5
      STOP
      END
```

## PROGRAM TALK2

| | |
|---|---|
| File: | TALK2 |
| Language: | Fortran 5 |
| Date: | October 19, 1982 |
| Author | K. Seelandt modified by J. Fletcher |
| Subject: | Speech Synthesis |
| Calling Sequence: | TALK2 Input file  Output file |

| ARGUMENT | TYPE | PURPOSE |
|---|---|---|
| OUTFILE | String | file name of resulting synthesized speech |
| OUT3 | String | file name of top phoneme choice from the recognition programs |
| JUNK, JUNK2 JUNK3, JUNK4 | Integers | time slice numbers from OUT3 file |
| P1,P2,P3,P4 | Integers | top phoneme choice for four consecutive time slices |
| SPEECHOUT | Integer Array | holds the corresponding speech time slices for chosen phonemes for OUTFILE |
| SPEECHIN | Integer Array | holds the corresponding speech time slices from file PHONEMES |

PURPOSE:

This program is used to create a speechfile that has been synthesized using the top phoneme choice from a recognition routine, for each time slice. The input file is a binary of these top choices.

DESCRIPTION:

LOCATION: FLETCHER.DR

ARGUMENT STRUCTURE:

OUTFILE = name of the speechfile to be created in a 7 character Hollerith string format

OUT3 = name of the input binary file holding the top choices in a 7 character Hollerith string format

PROGRAM USE:

This program is used to give an audible indication of how well the recognition routine worked in choosing the correct phoneme. The program uses a speechfile, PHONEMES, of phonemes created in record format. The phoneme value, from OUT3, for each time slice is then used to pick the corresponding speech phoneme and these are strung together to create a new speechfile in block format.

FILES CREATED:

OUTFILE: file name is user defined; holds the synthesized, digitized speech and is written in block format.

```
C       ****************************************
C       **                                    **
C       **      PROGRAM:   T A L K 2          **
C       **                                    **
C       **      WRITTEN BY:                   **
C       **              LT. KARL SEELANDT     **
C       **      MODIFIED BY:                  **
C       **              CAPT JAMES E. FLETCHER  **
C       **                                    **
C       **      DATE:   19 OCT 82             **
C       **                                    **
C       ****************************************
C       ****************************************
C
C       COMPILE IN FORTRAN V
C
C         This program is used to create digitized
c       speech using an OUT3 file such as the one created by
C       LISTER4 (ref. Seelandt's programs). Speech gener-
C       ation is done using a speech file of the single vector
C       phonemes stored individually as records corresponding to
C       their phoneme numbers.(This file is PHONEMES.)  The
C       top choice from the recognition routine for each time
C       slice is stored in OUT3.  This program then reads
C       that top choices and places the corresponding phoneme
C       in another speech file.  Since this is done sequentially,
C       a phoneme at a particular time slice corresponds directly
C       to the top phoneme choice for a particular time
C       slice listed in OUT3.
C
C
C
        IMPLICIT INTEGER (A-Z)  ;All variables are integers
        DIMENSION M1(7)
        INTEGER OUTFILE(7), SPEECHIN1(64), SPEECHOUT(256),
     ;  SPEECHIN2(64), SPEECHIN3(64), SPEECHIN4(64), OUT3(7)
        CALL IOF(2,M1,OUT3,OUTFILE,I1,M2,I2,I3,I4)
        CALL OPEN(1,"PHONEMES",1,128,IERO)
        IF(IERO .NE. 1) TYPE"ERROR OPENING FILE = ",IERO
        OPEN 2,OUT3                      ;Open channel to OUT3.
        DELETE OUTFILE                   ;Make sure there is no
        CALL CFILW(OUTFILE,2,IER)        ;and create specified file.
        CALL CHECK(IER)
        OPEN 3,OUTFILE
        COUNT = 0        ;Index used to increment block number in
                         ;speech file
C
C
  1     READ BINARY(2,END=20) JUNK, P1, JUNK2, P2,
     ;  JUNK3, P3, JUNK4, P4     ;OUT3 file is set up listing time
                         ;slice number and corresponding top
                         ;phoneme choice.
        CALL READRW(1,P1,SPEECHIN1,1,JER)       ;Read phoneme
        IF(JER .NE. 1) TYPE"JER = ",JER ;from phoneme
        CALL READRW(1,P2,SPEECHIN2,1,KER)       ;speech file
```

```
      IF(KER .NE. 1) TYPE"KER = ",KER
      CALL READRW(1,P3,SPEECHIN3,1,LER)
      IF(LER .NE. 1) TYPE"LER = ",LER
      CALL READRW(1,P4,SPEECHIN4,1,MER)
      IF(MER .NE. 1) TYPE"MER = ",MER
C
C
      DO 10 I=1,64
      SPEECHOUT(I) = SPEECHIN1(I)      ;Put the four phonemes
      SPEECHOUT(I+64) = SPEECHIN2(I)   ;into one array to
      SPEECHOUT(I+128) = SPEECHIN3(I)  ;create a block
      SPEECHOUT(I+192) = SPEECHIN4(I)  ;of speech.
 10   CONTINUE
      CALL WRBLK(3,COUNT,SPEECHOUT,1,NER)  ;Put block of speech
      IF(NER .NE. 1) TYPE"WRBLK ERROR: ",NER  ;in speech file.
      COUNT = COUNT + 1                ;Increment block counter
C
C
      GO TO 1          ;Start over and continue until come to
                       ;end of OUT3.
 20   WRITE(10,30) OUTFILE(1)
 30   FORMAT("FILE ",S13," HAS BEEN CREATED.")
      CALL RESET
      STOP
      END
```

## PROGRAM PLOTPHON

| | |
|---|---|
| File: | PLOTPHON |
| Language: | Fortran 5 |
| Date: | September 28, 1982 |
| Author | J. Fletcher |
| Subject: | phoneme recognition |
| Calling Sequence: | PLOTPHON |
| Subroutines used: | RDOUT1, RDOUT5, GRPH2 |

| ARGUMENT | TYPE | PURPOSE |
|---|---|---|
| INCHOICE | INTEGER | For choosing a routine to read either an OUT1 or OUT5 file |
| OUT1 | string | file name to be read |
| TITLE | string | Information to be printed with graph gives words spoken and speaker |
| IPHON | Integer | Phoneme number to be plotted |
| IPRINT | Integer | For choosing whether or not to print out amplitude file. |

PURPOSE:

This program is used to show graphically when a user chosen phoneme is picked as a top five choice in a speech-file. The graph is plotted on the Tektronix 4010-1 Graphic Terminal along with the phoneme number, the file used, the words spoken, and the speaker.

DESCRIPTION:

LOCATION: FLETCHER. DR

ARGUMENT STRUCTURE:

| | | |
|---|---|---|
| ICHOICE | = | 1 for choosing an OUT1 file or 0 for choosing an OUT5 file |
| OUT1 | = | the desired file to be used in Hollerith string format |
| TITLE | = | works spoken and speaker in Hollerith string format |
| TPHONE | = | 01 to 71 for OUT1 files or 001 to 071 for OUT5 files |
| IPRINT | = | 1 to print resulting amplitude file, 0 otherwise |

PROGRAM USE:

This program can use the recognition results for a given speechfile that has been run through either Seelandt's recognition routine, TRYDIST5 and LISTER4, or Martin's recognition routine, DRVR. The top five phoneme choices from Seelandt's routine are stored in file OUT1. File OUT5 is created by taking the file of Martin's top five choices and running it through program TOP5. OUT1 and OUT5 can have any file name and are referred to as OUT1 and OUT5 only to differentiate between the two types of files since they are formatted differently.

In order to load this program the load line must be:

```
RLDR PLOTPHON RDOUT1 RDOUT5 GRPH.LB
@FLIB@
```

Make sure you have linked to GRPH.LB which is in directory DP4F.

```
C       ****************************************
C       *                                      *
C       *                                      *
C       *  P R O G R A M   P L O T P H O N     *
C       *                                      *
C       *   WRITTEN BY:  JAMES E. FLETCHER     *
C       *                                      *
C       *   DATE:  28 SEP 82                   *
C       *                                      *
C       ****************************************
C
C
C       NOTE:  This program is for Portran 5.
C
C           PLOTPHON uses a file of the top five phoneme choices and
C       their associated scores for each time slice.
C       These files are the OUT1 files created by PHDIST and CHOICE5
C       and the OUT5 files created by DRVR, its associated subroutines
C       and by LISTTOP5.  PLOTPHON searches each time slice for a user
C       selected phoneme.  If one is found it's associated score is
C       placed in an array.  If one is not found for that time slice, a
C       zero is placed in the array.  The array of scores is then normal-
C       ized to a value betweenone and zero.  The scores are then plot-
C       ted on a graph with the phoneme score on the Y-axis and the time
C       slice number on X-axis.  This program utilizes three subroutines:
C       GRPH.LB, RDOUT1, RDOUT5.  Subroutine RDOUT1 reads OUT1 files,
C       searches for the desired phoneme and its score, normalizes that
C       score and returns the array of scores to the main program.
C       Subroutine RDOUT5 does everything RDOUT1does but reads OUT5
C       files instead of OUT1 files.  GRPH.LB is a packaged library
C       subroutine that plots graphs.  A pamphet is available for GRPH.LB.
C
C       NOTICE:  This program should used only on the TEKTRONIX 4010
C                teminal.
C
        DIMENSION SCARRAY(400), OUT1(13), TITLE(30), ISAID(50), ISPEAKR(13)
        IOPT = 0
        TYPE"THIS PROGRAM CAN USE TWO TYPES OF FILES,"
        TYPE"EITHER OUT1 OR OUT5 FILES."
  1     ACCEPT"DO YOU WANT AN OUT1 OR OUT5 FILE?(1=OUT1/0=OUT5)",ICHOICE
  2     ACCEPT"NAME OF FILE: "
        READ(11,3) OUT1(1)
  3     FORMAT(S13)
        ACCEPT"WORDS SPOKEN AND SPEAKER: "
        READ(11,4) TITLE(1)
  4     FORMAT(S30)
  5     CALL OPEN(2,OUT1,1,IER1);  This allows the computer to read the OUT1 file.
        CALL CHECK(IER1);  Check for an error during file opening.
        TYPE"NOTE: "
        TYPE"  TO CONTINUE AFTER GRAPH HAS BEEN DISPAYED,"
        TYPE"  HIT ANY KEY. "
        IF(IOPT .EQ. 2) GO TO 9
        IF(ICHOICE .EQ. 1) GO TO 7;  Phoneme formatted differently for
                                    ;OUT1.
        ACCEPT"ENTER PHONEME NUMBER (001 TO 071): "
        READ(11,6) IPHON
```

```
  6     FORMAT(I3)
        GO TO 9
  7     ACCEPT"ENTER PHONEME NUMBER (01 TO 71): "
        READ(11,8) IPHON
  8     FORMAT(A2)
  9     ACCEPT"DO YOU WANT TO PRINT OUT AMPL. FILE? (1=Y/0=N) ",IPRINT
        IMIN = 600;  No score should ever exceed this value.
        IMAX = 0
        ICONT = 0; Initializes flag for counting no. of times phoneme
C                  ; appears in OUT1 or OUT5 file.
        IF(ICHOICE .EQ. 1) CALL RDOUT1(MAX,MIN,ICONT,SCARRAY,IVECT,
     / IPHON,IMAX,IMIN)
        IF(ICHOICE .EQ. 0) CALL RDOUT5(MAX,MIN,ICONT,SCARRAY,IVECT,
     / IPHON,IMAX,IMIN)
  60    IF(ICONT .GT. 1) GO TO 65
        WRITE(10,61) IPHON, ICONT;f phoneme appears .LT. two times
  61    FORMAT(//,A2," APPEARS ONLY ",I1," TIME(S).");   the program is
        TYPE"PROGRAM UNABLE TO GENERATE AN OUTPUT.";   unable to plot
                                                      ;a graph.
        ACCEPT"TYPE ANY NUMBER TO CONTINUE. ",IPAUS
        GO TO 90
  65    CONTINUE
        DO 70 I=1,IVECT;  Perform the normalization
          IF(SCARRAY(I) .GT. 0) SCARRAY(I)=((SCARRAY(I)-MIN)/(MAX-MIN))
  70    CONTINUE
        IF(IPRINT .NE. 1) GO TO 82;  Skip if don't want array printed.
        IF(ICHOICE .EQ. 1) GO TO 74;  Phoneme formatted differently for
                                     ;OUT1.
        WRITE(12,71) IPHON
  71    FORMAT(" AMPLITUDES FOR PHONEME ",I3,//);  Header info for
        WRITE(12,75) IPHON                   ;printout
  75    FORMAT("  AMPLITUDES FOR PHONEME ",A2,//);  Header info for
        WRITE(12,76) OUT1(1)                 ;printout
  76    FORMAT("  FROM FILE ",S13,//);  Header info for print out.
        WRITE(12,80) (SCARRAY(I), I=1,IVECT);  Print hard copy of score
  80    FORMAT(5X,F6.3,5X,F6.3,5X,F6.3,5X,F6.3,5X,F6.3)     ;array
        WRITE(12,81) MAX, MIN
  81    FORMAT(//,5X,"MAX SCORE = ",F4.0,5X,"MIN SCORE = ",F4.0)
  82    CALL GRPH2("PHONEME AMPL.",1,SCARRAY,U,IVECT,0,XMIN,XMAX,0)
C
C       GRPH2 is the subroutine that plots the normalized scores on
C       a graph.
C
        IF(ICHOICE .EQ. 1) GO TO 84
        WRITE(10,83) IPHON, OUT1(1);  Header info for graph.
  83    FORMAT("AMPLITUDES FOR PHONEME # ",I3," FROM FILE ",S13)
        GO TO 86
  84    WRITE(10,85) IPHON, OUT1(1);  Header info for graph.
  85    FORMAT("AMPLITUDES FOR PHONEME # ",A2," FROM FILE ",S13)
  86    WRITE(10,87) TITLE(1);  Header info for graph.
  87    FORMAT("WORDS SPOKEN AND SPEAKER: ",S30)
        READ(11,88) IPAUSE
  88    FORMAT(S1);  Pause until user ready to clear sceen.
  90    CALL RESET
```

```
CALL ERS(1);  Subroutine to erase the screen.
CALL TDELAY(2);  Delay to allow time to clear screen before
                ;continuing.

TYPE"OPTIONS:"
TYPE"  1.  USE SAME FILE, BUT DIFFERENT PHONEME."
TYPE"  2.  GET DIFFERENT FILE, BUT USE SAME PHONEME.(MUST BE"
TYPE"      SAME TYPE FILE, EITHER OUT1 OR OUT5)"
TYPE"  3.  GET DIFFERENT FILE AND DIFFERENT PHONEME."
TYPE"  4.  END PROGRAM."
ACCEPT"ENTER OPTION NUMBER: ",IOPT
IF(IOPT .EQ. 1) GO TO 5
IF(IOPT .EQ. 2) GO TO 2
IF(IOPT .EQ. 3) GO TO 1
STOP
END
```

## SUBROUTINE RDOUT1

| | |
|---|---|
| File: | RDOUT1 |
| Language: | Fortran 5 |
| Date: | September 28, 1982 |
| Author: | J. Fletcher |
| Subject: | Reading output file of program CHOICE |
| Calling Sequence: | Call RDOUT1 (MAX, MIN, INCONT, SCARRAY, IVECT, IPHON, IMAX, IMIN) |

| ARGUMENT | TYPE | PURPOSE |
|---|---|---|
| MAX | Real | Maximum distance for specific phoneme in speechfile |
| MIN | Real | Minimum distance for specific phoneme in speechfile |
| ICONT | Integer | Count of the number of times specific phoneme shows up in output file |
| SCARRAY | Real Array | Array of distances for specific phoneme |
| IVECT | Integer | Time slice number where phoneme was chosen as a top five choice. |
| IPHON | Integer | Selected phoneme number for which distances are taken from output file |
| IMAX | Integer | Maximum distance for specific phoneme in speechfile |
| IMIN | Integer | minimum distance for specific phoneme in speechfile |

PURPOSE:

This subroutine is used in program PLOTPHON and reads the value from the output file of program CHOICE, OUT1.

DESCRIPTION:

Location: FLETCHER, DR

ARGEMENT STRUCTURE:

| | | |
|---|---|---|
| MAX | = | Used to compute the amplitude of the distances in main program. |
| MIN | = | Used to compute the amplitude of the distances in main program |
| ICONT | = | Used as a flag to notify user there are not enough distance values to plot a graph. |
| SCARRY | = | Passes distance values to main program: these will be converted to distance amplitudes. |
| IVECT | = | Used in graph routine to number points on X-axis. |
| IPHON | = | Can be an integer from 01 to 71; used to pick out corresponding distances in OUT1 file |
| IMAX | = | Initial maximum value passed to subroutine |
| IMIN | = | Initial minimum value passed to subroutine |

PROGRAM USE:

This subroutine is to be used with its main program, PLOTPHON. It is to be loaded in the fashion illustrated in the documentation for program plotphon. It will only read a file of the top five phoneme choices that has been formatted in the same manner as the OUT1 file created by program CHOICE.

```
      SUBROUTINE RDOUT1(MAX,MIN,ICONT,SCARRAY,IVECT,IPHON,IMAX,IMIN)
C
C     ****************************************
C     *                                      *
C     *   R O U T I N E    R D O U T 1       *
C     *                                      *
C     *   WRITTEN BY:  JAMES E. FLETCHER     *
C     *                                      *
C     *   DATE:  28 SEP 82                   *
C     *                                      *
C     ****************************************
C
C     NOTE:  This program is for Fortran 5.
C
C       Routine RDOUT1 is used by the main program PLOTPHON.
C     It uses the initialized flag, ICONT; the desired phoneme, IPHON,
C     and the initialized maximum and minimum scores, IMAX and IMIN.
C     It returns to the main program the updated floating point
C     maximum and minimum scores, MAX and MIN; the updated flag,
C     ICONT; the number of time slices in the OUT1 file, IVECT, and
C     the array of scores.  RDOUT1 goes to the OUT1 file and searches
C     for the desired phoneme. Upon   locating it, it places its
C     score in the array at the corresponding time slice number.
C     Every time it finds the desired phoneme it also increments
C     ICONT. It also checks to see if the score is a new
C     maximum or a new minimum.
C
C
      DIMENSION SCARRAY(400)
      DO 8 I=1,400
        SCARRAY(I) = 0
 8    CONTINUE
 10   READ(2,20,END=60) IVECT, IPHON1, IVAL1, IPHON2, IVAL2, IPHON3,
    / IVAL3, IPHON4, IVAL4, IPHON5, IVAL5 ;  Read in phonemes & scores
 20     FORMAT(5X,I4,3X,A2," ",I3,2X,A2," ",I3,2X,A2," ",I3,
    /   2X,A2," ",I3,2X,A2," ",I3)
C     These if statements check to see if any of the phonemes match
C     the desired phoneme.  If a match is found, its score becomes
C     ISCORE and ICONT is incremented.
C
        IF(IPHON .EQ. IPHON1) GO TO 30
        IF(IPHON .EQ. IPHON2) GO TO 31
        IF(IPHON .EQ. IPHON3) GO TO 32
        IF(IPHON .EQ. IPHON4) GO TO 33
        IF(IPHON .EQ. IPHON5) GO TO 34
        GO TO 10
 30     ISCORE = IVAL1
        ICONT = ICONT + 1
        GO TO 40
 31     ISCORE = IVAL2
        ICONT = ICONT + 1
        GO TO 40
 32     ISCORE = IVAL3
        ICONT = ICONT + 1
        GO TO 40
 33     ISCORE = IVAL4
```

```
        ICONT = ICONT + 1
        GO TO 40
34      ISCORE = IVAL5
        ICONT = ICONT + 1
40      IF(ISCORE .GT. IMAX) IMAX = ISCORE;  Sets IMAX to the highest
                                             ;score
        IF(ISCORE .LT. IMIN) IMIN = ISCORE;  Sets IMIN to the lowest
                                             ;score.
        SCORE = FLOAT(ISCORE);  Change to floating point
        MAX = FLOAT(IMAX)      ;numbers
        MIN = FLOAT(IMIN)
        SCARRAY(IVECT) = SCORE;  Put score into array.
      GO TO 10;  Continue until end of file is reached.
60    RETURN
      END
```

SUBROUTINE RDOUT5

File: RDOUT5

Language: Fortran 5

Date: September 28, 1982

Author: J. Fletcher

Subject: Reading output file of program TOP5

Calling Sequences: Call RDOUT5 (MAX, MIN, ICONT, SCARRAY, IVECT, IPHON, IMAX, IMIN)

| ARGUMENT | TYPE | PURPOSE |
|---|---|---|
| MAX | Real | Maximum distance for specific phoneme in speechfile |
| MIN | Real | Minimum distance for specific phoneme in speechfile |
| ICONT | Integer | Count of the number of times specific phoneme shows up in output file |
| SCARRAY | Real Array | Array of distances for specific phoneme |
| IVECT | Integer | Time slice number where phoneme was chosen as a top five choice |
| IPHON | Integer | Selected phoneme number for which distances are taken from output file |
| IMAX | Integer | Maximum distance for specific phoneme in speechfile |
| IMIN | Integer | Minimum distance for specific phoneme in speechfile |

PURPOSE:

This subroutine is used in program PLOTPHON and read the values from the output file of program TOP5, OUT5.

DESCRIPTION:

Location: FLETCHER. DR

ARGUMENT STRUCTURE:

| | | |
|---|---|---|
| MAX | = | Used to compute the amplitude of the distance in main program |
| MIN | = | Used to compute the amplitude of the distance in main program |
| ICONT | = | Used as a flag to notify user there are not enough distance values to plot a graph |
| SCARRAY | = | Passes distance values to main program; there will be converted to distance amplitudes |
| IVECT | = | Used in graph routine to number points on X-axis |
| IPHON | = | Can be an integer from 02 to 71; used to pick out corresponding distances in OUT1 file |
| IMAX | = | Initial maximum value passed to subroutine |
| IMIN | = | Initial minimum value passed to subroutine |

PROGRAM USE:

The subroutine is to be used with the main program, PLOTPHON. It is to be loaded in the same fashion as outlined in the documentation for program PLOTPHON. It performs the same function as subroutine RDOUT1, but will only read a file formatted in the same manner as the OUT5 file created by program TOP5.

```
      SUBROUTINE RDOUT5(MAX,MIN,ICONT,SCARRAY,IVECT,IPHON,IMAX,IMIN)
C
C     ***************************************
C     *                                     *
C     *     R O U T I N E   R D O U T 5     *
C     *                                     *
C     *      WRITTEN BY:  JAMES E. FLETCHER *
C     *                                     *
C     *      DATE:  28 SEP 82               *
C     *                                     *
C     ***************************************
C
C     NOTE:  This program is for Fortran 5.
C
C         Routine RDOUT5 is used by the main program PLOTPHON.
C     It uses the initialized flag, ICONT; the desired phoneme, IPHON,
C     and the initialized maximum and minimum scores, IMAX and IMIN.
C     It returns to the main program the updated flag, ICONT; the
C     number of time slices, IVECT, in the OUT5 file and the array
C     of scores.  RDOUT5 goes to the OUT5 file and searches for the
C     desired phoneme.  Upon locating it, it places its score in the
C     array at the corresponding time slice number.  Every time it
C     finds the desired phoneme it also increments ICONT.  It also
C     checks to see if the score is a new maximum or a new minimum.
C
C
      DIMENSION SCARRAY(400), ISAID(50), ISPEAKR(13)
      DO 8 I=1,400
        SCARRAY(I) = 0
  8   CONTINUE
C      The following seven READ statements just reads off the OUT5
C     file header to get you into the main body of the file.
C
      READ(2,11) ISAID(1)
 11   FORMAT(15X,"SENTENCE SPOKEN : ",S50,/)
      READ(2,12) ISPEAKR(1)
 12   FORMAT(19X,"SPEAKER WAS : ",S13,/)
      READ(2,13) IDATE
 13   FORMAT(15X,"THE DATE IS--",2I3,I5)
      READ(2,14) IHOUR, IMIN, ISEC
 14   FORMAT(15X,"THE TIME IS--",3I3,//)
      READ(2,15)
      READ(2,16)
      READ(2,17)
C*******
 15   FORMAT(4X,"VECTOR",4X,"FIRST",3X,"SECOND",4X,"THIRD",
     / 3X,"FOURTH",4X,"FIFTH",7X,"SCALE",6X,"VECTOR")
 16   FORMAT(4X,"NUMBER",3X,"CHOICE",3X,"CHOICE",3X,"CHOICE",
     / 3X,"CHOICE",3X,"CHOICE",6X,"FACTOR",6X,"ENERGY")
 17   FORMAT(4X,"******",2X,"*******",2X,"*******",2X,"*******",
     / 2X,"*******",2X,"*******",5X,"*******",5X,"*******",/)
 10   READ(2,20,END=60) IVECT, IPHON1, IVAL1, IPHON2, IVAL2, IPHON3,
     / IVAL3, IPHON4, IVAL4, IPHON5, IVAL5;  Read in phonemes and
                                             ;scores.
 20     FORMAT(5X,I4,3X,I3," ",I3,2X,I3," ",I3,2X,I3," ",I3,
```

```
     /    2X,I3," ",I3,2X,I3," ",I3)
C        These IF statements check to see if any of the phonemes match
C        the desired phoneme.  If a match is found, its score becomes
C        ISCORE and ICONT is incremented.
C
          IF(IPHON .EQ. IPHON1) GO TO 30
          IF(IPHON .EQ. IPHON2) GO TO 31
          IF(IPHON .EQ. IPHON3) GO TO 32
          IF(IPHON .EQ. IPHON4) GO TO 33
          IF(IPHON .EQ. IPHON5) GO TO 34
          GO TO 10
  30      ISCORE = IVAL1
          ICONT = ICONT + 1
          GO TO 40
  31      ISCORE = IVAL2
          ICONT = ICONT + 1
          GO TO 40
  32      ISCORE = IVAL3
          ICONT = ICONT + 1
          GO TO 40
  33      ISCORE = IVAL4
          ICONT = ICONT + 1
          GO TO 40
  34      ISCORE = IVAL5
          ICONT = ICONT + 1
  40      IF(ISCORE .GT. IMAX) IMAX = ISCORE;  Sets IMAX to the highest
                                                ;score.
          IF(ISCORE .LT. IMIN) IMIN = ISCORE;  Sets IMIN to the lowest
                                                ;score.
          SCORE = FLOAT(ISCORE);  change score to floating point
          MAX = FLOAT(IMAX)              ;number.
          MIN = FLOAT(IMIN)
          SCARRAY(IVECT) = SCORE;  Put SCORE into array.
        GO TO 10;  Continue until end of file is reached.
  60    RETURN
        END
```

## PROGRAM GENOUT3

| | |
|---|---|
| File: | GENOUT3 |
| Language: | Fortran 5 |
| Date: | November 2, 1982 |
| Author: | J. Fletcher |
| Subject: | Generation of top choice file |
| Calling Sequence: | CHNGFILE |

| ARGUMENT | TYPE | PURPOSE |
|---|---|---|
| IFILE | String | file name of top 5 phoneme distances |
| IHEADER | Integer Array | leader information of top 5 file |
| ILAST | Integer | number of time slices in speech-file |
| IRRAY | Integer Array | holds one block information from top 5 file |
| ISLICE | Integer | time slice number of top 5 phoneme choice |
| IVAL1-IVAL12 | Integer | unused values from top 5 file |
| IPHON | Integer | top phoneme choice for a time slice |

DESCRIPTION:

Location: FLETCHER. DR

ARGUMENT STRUCTURE:

IFILE = name of top 5 phoneme distances from DRVR: formatted in 13 character Hollerith string

PROGRAM USE:

This program creates an OUT3 file to be used as an input file for program TALK2. It uses a file created by DRVR when the option, choose 5 best distances, is used.

FILES CREATED:

| | |
|---|---|
| TEMFILE | = contains everything that is in IFILE except the header block: written in block format |
| OUT3 | = contains the time slice and corresponding top phoneme choice: written in WRITE BINARY format |

```
C       ########################################
C       **                                    **
C       **    PROGRAM:  G E N O U T 3         **
C       **                                    **
C       **    BY:  CPT JAMES E. FLETCHER      **
C       **                                    **
C       **    DATE:    2 NOV  82              **
C       **                                    **
C       ########################################
C
C       NOTE: THIS IS FOR FORTRAN 5.
C
C       This program takes the file created by DRVR for the five best
C       distances when a speech file is banged against a phoneme file and
C       creates an OUT3 file.  The OUT3 file is a list of the time slices in
C       a speech file and its corresponding top phoneme choice.
C       The OUT3 file, once created, can now be run in program TALK2 to
C       resynthesize the speech recognized by DRVR.  This gives an audible
C       indication of how well DRVR did at recognizing speech.
C
C
C
        DIMENSION IARRAY(256), IHEADER(256)
        ACCEPT"FILE NAME OF TOP 5 DISTANCES FROM DRVR: "
        READ(11,5) IFILE
  5     FORMAT(S13)
        CALL OPEN(1,IFILE,1,IER1)
        CALL CHECK(IER1)
        CALL FOPEN(2,"TEMFILE")          ; File used to temporarily store
                                 ;the top 5 choices minus the header block.
        CALL RDBLK(1,0,IHEADER,1,IER0)
        CALL CHECK(IER0)
        ILAST = IHEADER(41)      ;Value for # of time slices in speech file.
        TYPE"ILAST = ",ILAST
        ISTART1 = 1      ; Start block for reading top 5 file.
        ISTART2 = 0      ;  Start block for writing to TEMFILE.
  10    CALL RDBLK(1,ISTART1,IARRAY,1,IER2)    ; Read top 5 file.
        IF(IER2 .EQ. 9) GO TO 20
        ISTART1 = ISTART1 + 1    ; Increment to next block.
        CALL WRBLK(2,ISTART2,IARRAY,1,IER3)      ; Write to TEMFILE.
        CALL CHECK(IER3)
        ISTART2 = ISTART2 + 1    ;Increment to next block.
        GO TO 10
  20    CALL RESET
C
C
        TYPE"TEMFILE HAS BEEN CREATED."
        CALL FOPEN(2,"TEMFILE")
        CALL DFILW("OUT3",IER5)
        IF(IER5 .NE. 1) TYPE "IER5 = ",IER5
        CALL FOPEN(3,"OUT3")
C       A string of values is read in to get the time slice number
C       (ISLICE) and phoneme number (IPHON).  Reading in the string
C       of values after IPHON sets you up to read in the next time
```

```
C       slice number.
  30    READ BINARY(2,END=40) ISLICE, IVAL1, IVAL2, IVAL3, IPHON, IVAL4,
     /  IVAL5, IVAL6, IVAL7, IVAL8, IVAL9, IVAL10, IVAL11, IVAL12
        WRITE BINARY(3) ISLICE, IPHON   ;Write time slice & its top
                                        ; phoneme to OUT3.
        IF(ISLICE .EQ. ILAST) GO TO 40  ; Check for last time slice number (In case
                                        ; top 5 file overruns speech file.
        GO TO 30
  40    CALL RESET
        CALL DFILW("TEMFILE",IER4)      ; Delete temporary file holding
                                        ; top 5 choices.
        CALL CHECK(IER4)
        TYPE"TEMFILE HAS BEEN DELETED."
        TYPE" "
        TYPE"OUT3 HAS BEEN CREATED."
        STOP
        END
```

PROGRAM CHNGFILE

File: CHNGFILE

Language: Fortran 5

Date: November 1, 1982

Author: J. Fletcher

Subject: Spectrum file for program DRVR

Calling Sequence: CHNGFILE

| ARGUMENT | TYPE | PURPOSE |
|---|---|---|
| IFILE | String | identified spectrum file to be used |
| INEWFIL | String | identifies file name for altered spectrum file |
| IHEADER | Integer Array | holds header information of spectrum file |
| ICOMP | Integer | number of frequency components per time slice |
| IHEADER (56) | Integer | number of time slices in spectrum file |
| SPCHARY | Real Array | holds 2 blocks of spectral components |
| ILENVAL | Integer | number of time slices in spectrum file; user can alter IHEADER (56) for using SPENPLOT |
| ILENGTH | Integer | time slice length of phonemes |
| ARRAY1 | Real Array | altered spectrum of speechfile |

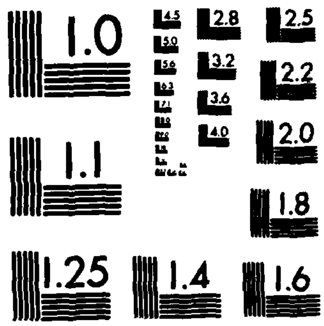
PURPOSE:

This program alters the original spectrum of a speechfile that has had a 64 point DFT performed on it. This allows DRVR to perform 2 to 4 vector phoneme recognition on a speech file.

AD-A144 562 PERFORMANCE IMPROVEMENTS OF THE PHONEME RECOGNITION ALGORITHM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING J E FLETCHER JUN 84 2/2

UNCLASSIFIED AFIT/GE/EE/84J-2 F/G 17/2 NL

END


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

DESCRIPTION:

Location: FLETCHER. DR

ARGUMENT STRUCTURE:

| | | |
|---|---|---|
| IFILE | = | name of original spectrum file created by DRVR: formatted as a 13 character Hollerith string |
| INEWFIL | = | name of spectrum file alter by CHNGFILE in 13 character Hollerith string format |
| ICOMP | = | integer that is a multiple of 32 |
| ILENVAL | = | integer value equal to original number of time slices when SPENPLOT (ref 7) is to be used |
| ILENGTH | = | integer value of 2 or 4 depending on phoneme length used as prototype |

PROGRAM USE:

This program is to be used only on a spectrum file formatted as the spectrum file created by program DRVR (Ref 6). In this spectrum there is a one block header attached to the file and every thirty-second element is the energy of that time slice. This program is to be used on both the spectrum of the speechfile and the phoneme speechfile.

FILES CREATED:

INEWFIL: file name is user defined: holds altered spectrum components and is written in block format with a one block header attached and each thirty-second component is a constant 1.0.

```
C       ******************************************
C       **                                      **
C       **   PROGRAM:   C H N G F I L E         **
C       **                                      **
C       **   BY:  CPT JAMES E. FLETCHER         **
C       **                                      **
C       **   DATE:   1 NOV  82                  **
C       **                                      **
C       ******************************************
C
C       NOTE:  THIS PROGRAM IS FOR FORTRAN 5.
C
C          Basically, the purpose of this program is to alter the spectrum
C       files created by Martin's DRVR program in order to perform phoneme
C       recognition on speech files using different length phoneme templates.
C       This program performs two tasks:
C               1.  It takes the spectrum file of the phoneme template and
C               sets the energy value to the constant 1.0, this is the
C               first word in each group of 32 words making up a time slice's
C               frequency components.  It a so changes the information in
C               the header block so that the  correct number of components are
C               banged against each other(i.e. for a 4 vector phoneme, a group
C               of 128 components must be banged together).  The other
C               value to be changed in the header block is the number of
C               vectors in the new spectrum file (this depends on the number
C               of vectors per phoneme).
C
C               2.  It takes the spectrum file of the speech file and performs
C               the same task as with a phoneme spectrum file, plus it
C               expands the spectrum file.  This expansion is based on the
C               algorithm used when performing distance measuring with a
C               speech file and a phoneme template set as in Seelandt's
C               program.  The speech file is divided into groups of vectors
C               where the number of vectors per group is equal to the number
C               of vectors per phoneme template.  There is also an overlap
C               with the groups such that the beginning vector of a one
C               group of vectors was the second vector of the last group
C               of vectors.
C
C       The resulting spectrum files can then be run in DRVR, making DRVR
C       perform phoneme recognition using 1, 2, and 4 vector phonemes as
C       templates.
C
C
        DIMENSION IHEADER(256), SPCHARY(256), ARRAY1(640), IFILE(13), INEWFIL(1)
        TYPE"IS THE FILE TO BE MODIFIED FROM"
        TYPE"A SPEECH FILE OR A PHONEME FILE?"
        ACCEPT"(TYPE 0 FOR SPEECH OR 1 FOR PHONEME): ",IANS
        TYPE " "
        TYPE"ENTER NAME OF SPECTRUM FILE"
        IF(IANS .EQ. 1) ACCEPT"TO BE USED AS A PHONEME FILE: "
        IF(IANS .EQ. 0) ACCEPT"TO BE USED AS AN OBSERVATION FILE: "
        READ(11,5) IFILE(1)
  5     FORMAT(S13)
```

```
      CALL OPEN(1,IFILE,1,IER1)
      CALL CHECK(IER1)
      ACCEPT"ENTER NEW FILE NAME: "   ; File name for new spectrum file.
      READ(11,5) INEWFIL(1)
      ACCEPT"ENTER VECTOR LENGTH OF PHONEMES: ",ILENGTH
      CALL FOPEN(2,INEWFIL)   ;Opens channel to new spectrum file.
      ISTART = 1              ; Beginning block of freq. components in old file
      ACCEPT"NO. COMPONENTS/ VECTOR IN NEW FILE: " ,ICOMP
      ICONT = 1               ; For counting # of 32 point vectors.
      IBLK = 1                ; Starting block # for new file.
C
C
      CALL RDBLK(1,0,IHEADER,1,IER0)  ; Read header block.
      CALL CHECK(IER0)
      IHEADER(57) = ICOMP     ; Change # of freq. components
      ICHECK = (IHEADER(56) - (ILENGTH - 1)) * ILENGTH   ; Checks for
                   ;correct # of 32 point vectors in new file.
      IF(IANS .EQ. 1) GO TO 9
      IF(IANS .EQ. 0) IHEADER(56) = IHEADER(56) - (ILENGTH - 1)  ; # of
                              ; ICOMP point vectors is speech spectrum
  9   IF(IANS .EQ. 1) IHEADER(56) = IHEADER(56)/ILENGTH  ;# of ICOMP
                              ; point vectors in phoneme spectrum.
      TYPE"FILE LENGTH (IN VECTORS)= ",IHEADER(56)
      ACCEPT"DO YOU WANT TO CHANGE THIS VALUE?(1=Y/0=N)",IANSA  ; Can be
      IF(IANSA .EQ.0) GO TO 11        ; changed for plotting spectrum using
              ;SPENPLOT program (must also change ICOMP to 32).
      ACCEPT"ENTER FILE LENGTH VALUE: ",ILENVAL  ;Should be the same
      IHEADER(56) = ILENVAL           ; as ICHECK.
 11   CALL WRBLK(2,0,IHEADER,1,IER4)  ; Write header block to new spectrum.
      CALL CHECK(IER4)
C
C
 10   CALL RDBLK(1,ISTART,SPCHARY,2,IER2)  ;Read 2 blocks of spectral components.
      IF(IER2 .EQ. 9) GO TO 55        ;If the end of file, end the program.
      IJMP = 1        ;To increment for each 32nd freq. component.
      INDEX = 0               ; Index in the old spectrum array.
      INCR = 0                ;Index in the new spectrum array.
      DO 20 I=1,8
         SPCHARY(IJMP) = 1.0  ;Change energy value to a constant.
 20      IJMP = IJMP + 32     ; Go to next energy value.
      IF(IANS .EQ. 0) GO TO 25
      CALL WRBLK(2,IBLK,SPCHARY,2,IER3)  ;For phoneme spectrum, write blocks
      IF(IER3 .NE. 1) TYPE"WRBLK ERROR = ",IER3  ; to new file.
      IBLK = IBLK + 2 ; Increment to next two blocks for reading.
      ISTART = ISTART + 2     ;Return to read 2 more blocks.
      GO TO 10
C
C
 25   DO 30 II=1,4    ;Do for 4 groups of vectors.
         DO 40 J=1,ILENGTH   ;Do for # of vectors per phoneme
            DO 50 JJ=1,32    ;create one group.
 50            ARRAY1(JJ+INCR) = SPCHARY(JJ+INDEX)  ;Put freq. components
                                                    ;in new array.
            ICONT = ICONT + 1 ;Count # of vectors created in new file.
```

```
            IF(ICONT .EQ. ICHECK) GO TO 55   ; Check to make sure correct
                                  ;amount of vectors have been created.
            INCR = INCR + 32  ;Jump over last set of components.
            INDEX = INDEX + 32        ; Increment to next 32 components.
40       CONTINUE
30       INDEX = II * 32   ; Set to begin reading in components for next
                                  ; group of vectors.
      CALL WRBLK(2,IBLK,ARRAY1,ILENGTH,IER3)  ; Read in # of blocks
      IF(IER3 .NE. 1) TYPE"WRBLK ERROR = ",IER3  ; created by DO loops.
      IBLK = IBLK + ILENGTH  ; Increment to next set of blocks.
      ISTART = ISTART + 1   ; Increment 1 block to read in next two blocks.
      GO TO 10                    ; Start over again.
55    CALL RESET
      STOP
      END
```

# PROGRAM SGRAM

| | |
|---|---|
| **File:** | SGRAM |
| **Language:** | Fortran 5 |
| **Date:** | October 20, 1982 |
| **Author:** | K. Seelandt, modified by J. Fletcher |
| **Subject:** | voice spectrograms |
| **Call Sequence:** | SGRAM |

| ARGUMENT | TYPE | PURPOSE |
|---|---|---|
| SFREQ | Real | value set for the sampling frequency |
| IHEIGHT | Integer | height of printed characters per frequency component |
| IWIDTH | Integer | width of printed character per frequency component |
| IPRE | Integer | width for performing preemphasis |
| IDB | Integer | number of dB to preemphasis |
| FREQ | Real | starting frequency of preemphasis |
| ISIZE | Integer | number of samples per DFT |
| ENTHRES | Real | energy threshold for normalization |
| IFILENAM | String | name of the speechfile to be used |
| ISTART | Integer | First block in speechfile to be used in spectrogram |
| ILAST | Integer | last block in speechfile to be used in spectrogram |
| IN | String | title of spectrogram |
| RENERGY | Real Array | holds energy for each spectral component in sentence |

| | | |
|---|---|---|
| DTARAY | Real Array | holds frequency components to be used in DFT |
| IDSP | Integer Array | holds a block of components read from speechfile |
| B | Real Array | holds windowed frequency components |
| IBI | Integer Array | holds integer values of frequency components after DFT is performed |

PURPOSE:

This program is used to plot a spectrogram of a speechfile using a 64 point DFT and either Hamming a Rectangular window.

DESCRIPTION:

Location: FLETCHER. DR

ARGUMENT STRUCTURE:

| | |
|---|---|
| IHEIGHT | = 1 to plot normal size spectrogram |
| IWIDTH | = 1 to plot normal size spectrogram |
| IFILENAM | = desired speechfile in a 13 character Hollerith string format |
| ISTART | = 0 to start at beginning of speechfile |
| ILAST | = usually last block of the speechfile must not exceed file block size |
| ISIZE | = 64 for 64 point DFT |
| ENTRES | = value can be set by user; 10 was used exclusively in this thesis |
| IN | = 79 character Hollerith string format; selected by user |

PROGRAM USE:

This program can only be used with the Printronix printer. The speechfile used should be in block format no restrictions on size. Be sure speechfile is digitized speech and not a spectrum file. The loading is as follows:

RLDR SGRAM BYTEOUT EDFT.LB @FLIB@
BYTEOUT.RB is located on Fletcher's tape #1.

FILES CREATED:

| | |
|---|---|
| TENERGY: | contains component numbers and the total energy in the speechfile for each spectral component |
| IENERGY: | contains the vector number and the total energy for each vector |
| DCOMPS: | contains the DFT frequency components of the vectors each vector is stored as a record |

```
C       ##########################################################
C       ##########################################################
C       **                                                      **
C       **      PROGRAM:  S G R A M                             **
C       **                                                      **
C       **              DATE: 20 OCT 82                         **
C       **                                                      **
C       **                      BY: LT. KARL SEELANDT           **
C       **                                                      **
C       **              REVISED BY: CAPT JAMES E. FLETCHER      **
C       **                                                      **
C       ##########################################################
C       ##########################################################
C
C       COMPILE AS A FORTRAN V PROGRAM
C
C******     This program was developed by LT. MARK FELKEY during
C       his thesis at AFIT.  This program creates a spectrogram of
C       a speech file by finding the DFT of a set of digitized
C       speech samples, then preemphasizing the high frequency
C       components of the resultant frequency vector (to obtain
C       spectrograms that look as much as possible like the one's
C       obtained by Potter, Kopp and Green in Visible Speech).
C       The background noise is effectively reduced by threshold-
C       ing the energy of each frequency vector (total energy is
C       equal to: square-root of the sum of the squares of individual
C       frequency components).  This threshold value is obtained by
C       adjusting the value to remove as much unwanted signal as
C       is possible, while not degrading the recognition results.
C       Finally, the spectrogram is printed out on the printronix
C       printer using a gray-scale plotting technique.
C       This program has been modified to use either Hamming windowing
C       or rectangular windowing.
C
C INITIALIZE VARIABLES
C
        DIMENSION DTARAY(256),ISYMBOL1(10),ISYMBOL6(10),IDATE(3)
        DIMENSION B(256),IBI(128),IDSP(256),RENERGY(64)
        DIMENSION IFILENAM(13),IN(79)
        DIMENSION ISYMBOL2(10),ISYMBOL3(10),ISYMBOL4(10),ISYMBOL5(10)
        COMMON/BLK/ISYMBOL1,ISYMBOL2,ISYMBOL3,ISYMBOL4,ISYMBOL5,ISYMBOL6
        COMPLEX DTARAY
C
C CRITICAL VARIABLES DEFINED. SOME OF THESE MAY BE OPERATOR ADJUSTED BY
C OPTIONALLY COMPILING THE PROGRAM WHICH COMPILES THE STATEMENTS PRECEEDED BY
C AN 'X' IN COLUMN 1---(USE FORT/X SGRAM).
C
        ISAMP=1   ;COUNTS THE NUMBER OF VECTORS CREATED.
        ICOUNT=1   ;VARIABLE THAT KEEPS TRACK OF WHEN 10 VECTORS WERE OUTPUT.
        ICHAN=3  ;I/O CHANNEL FOR SGRAM OUTPUT (USED IN SUBROUTINE CALL)
        SFREQ=8000.0  ;SAMPLING FREQUENCY
        IHEIGHT=1 ;HEIGHT OF SGRAM CHARACTERS
        IWIDTH=1  ;WIDTH OF SGRAM CHARACTERS
        IPRE=1    ;SIGNAL TO DO PREEMPHASIS
```

```
      IDB=6     ;NUMBER OF DB'S TO PREEMPHASIZE
      FREQ=500.0;STARTING FREQ FOR PREEMPHASIS
      IPRESS=2  ;SIGNAL TO NOT COMPRESS VECTORS
      ISIZE=64  ;NUMBER OF SAMPLES PER DFT
      ENTHRES=90.0 ;ENERGY THRESHOLD FOR NORMALIZATION
C
C PLOTTER CHARACTERS USED BY THE PRINTRONIX PLOTTER FUNCTION.
C
      IPLOT=005K  ;PLOT COMMAND
      ILF=012K    ;PRINT LINE OF DATA JUST SENT.
      IDASH=177K  ;DASH USED FOR SCALE ON SGRAM
      IBLANK=0    ;EMPTY CHARACTER USED TO MAKE SURE COMPLETE WORD IS OUTPUT
C
C SPECTROGRAM SYMBOLS THAT MAKE UP THE GRAY-SCALE CHARACTERS. EACH SYMBOL
C DEFINES 1 OF 6 DOT LINES FOR 10 LEVELS OF INTENSITY.
C
      DATA ISYMBOL1/100K,100K,100K,122K,122K,122K,122K,166K,177K,177K/
      DATA ISYMBOL2/100K,122K,166K,166K,177K,177K,177K,177K,177K,177K/
      DATA ISYMBOL3/100K,100K,100K,100K,100K,122K,133K,133K,133K,177K/
      DATA ISYMBOL4/100K,100K,100K,122K,122K,122K,122K,166K,177K,177K/
      DATA ISYMBOL5/100K,122K,166K,166K,177K,177K,177K,177K,177K,177K/
      DATA ISYMBOL6/100K,100K,100K,100K,100K,122K,133K,133K,133K,177K/
C
C INPUT CONTROL VARIABLES, OPEN FILES, & PRINT HEADING ON SGRAM.
C
      CALL FOPEN(1,"TENERGY")  ;FILE TO INDICATE RELATIVE TOTAL
                               ;ENERGY OF SENTENCE.
      CALL FOPEN(4,"IENERGY") ;FILE FOR INDIVIDUAL ENERGY OF EACH VECTOR.
C
C  INPUT NAME OF FILE THAT THE SGRAM WILL BE MADE FROM=FILENAM
C
      ACCEPT"ENTER FILENAME TO BE SGRAM' ED :  "
      READ(11,669)IFILENAM(1)
  669 FORMAT(S13)
C
C
C
      CALL OPEN(5,IFILENAM,1,512,IER) ;FILE THAT CONTAINS DIG. SPEECH
      IF(IER.NE.1) TYPE "ERROR ON OPEN,IER=",IER
        ;CHECK TO SEE IF SGRAM IS DESIRED.
      ACCEPT "CONSTRUCT SPECTROGRAM? (1=YES/2=NO):",ISPEC
      IF (ISPEC.NE.1) GOTO 591
      ACCEPT "SEND SGRAM TO PRINTER? (Y=1,N=2):",IREPLY
      IF (IREPLY.NE.1) GO TO 580
      CALL FOPEN(3,"$LPT")
      GO TO 590
        ;
 580  CALL FOPEN(3,"SYMBOLS","B") ;IF SGRAM ISN'T SENT TO PRINTER,STORE
                     ;CHARACTERS IN THIS FILE FOR OUTPUT LATER.
 590  CONTINUE
X     ACCEPT "HEIGHT OF SYMBOLS OF SGRAM?",IHEIGHT
X     ACCEPT "WIDTH OF SYMBOLS OF SGRAM?",IWIDTH
 591  ACCEPT "FIRST BLOCK TO READ=",ISTART
      ACCEPT "LAST BLOCK TO READ=",ILAST
```

```
X       ACCEPT "NUMBER OF SAMPLES PER VECTOR (POWER OF 2):",ISIZE
        ACCEPT"DO YOU WANT A HAMMING WINDOW?(1=Y/0=N) ",IHAM
        ISIZEH=ISIZE/2 ;NUMBER OF COMPONENTS PER VECTOR
        OPEN 2,"DCOMPS",LEN=ISIZE ; STORES THE FREQ. COMPS. OF EACH VECTOR
        ACCEPT "ENERGY THRESHOLD=",ENTHRES
        IBYTE=999  ;INITIALIZE SUBROUTINE BYTEOUT WITH THIS VARIABLE
        CALL BYTEOUT(ICHAN,IBYTE)
        IF (ISPEC.NE.1) GOTO 599
        ;
        ;PRINT HEADING ON SPECTROGRAM.
        ;
        ACCEPT"ENTER TITLE OF SCRAM (79 CHAR. OR LESS)(15) "
        READ(11,69)IN(1)
  69    FORMAT(S79)
        WRITE(3,70)IN(1)
  70    FORMAT(1X," "//" **********",S79,"********** "//" ")
C
        CALL DATE(IDATE,IER)
        IF(IER.NE.1) TYPE "ERROR ON DATE,IER=",IER
        CALL FGTIME(IHOUR,IMIN,ISEC)
        WRITE(3,592) (IDATE(I),I=1,3)
 592    FORMAT(1X,"DATE:",2I3,I5)
        WRITE(3,593) IHOUR,IMIN,ISEC
 593    FORMAT(1X,"TIME:",3I3)
        WRITE(3,595) ISIZE,ISTART,ILAST
 595    FORMAT(1X,"NO. OF SAMPLES=",I3,"  FIRST BLOCK=",I3," LAST BLOCK=",
     *  I3)
        IF(IPRE.EQ.1) WRITE(3,596) IDB
 596    FORMAT(1X,"HIGH FREQUENCIES PREEMPHASIZED, DB/OCTAVE=",I3)
        IF(IPRE.EQ.1) WRITE(3,597) FREQ
 597    FORMAT(1X,"PREEMPHASIS STARTS AT FREQ:",F5.0)
        WRITE(3,601) ENTHRES
 601    FORMAT(1X,"ENERGY THRESHOLD=",F6.1)
C
C
        JOE=1
        WRITE(3,769)JOE
  769   FORMAT(" "//" ***************************
     :  **************",I1,"********************
     :  ******************** "//" ")
C
          ;
          ;
          ;
          ;THIS SECTION OF THE PROGRAM READS IN DIG.SPEECH AND CREATES SCRAM.
          ;
          ;
 599    CONTINUE
        DO 605 I=1,ISIZEH
 605    RENERGY(I)=0.0 ;CLEAR ARRAY THAT ADDS UP ENERGY OF SENTENCE
        ;
        ;MAIN LOOP OF PROGRAM.
        ;
        DO 1000 IN=ISTART,ILAST
```

```
        ;INPUT SPEECH AND PROCESS IT.
        CALL RDBLK(5,IN,IDSP,1,IER)
        IF(IER.NE.1) TYPE "ERROR ON RDBLK,IER=",IER
        J1=0
 600    CONTINUE
        ;FORM COMPLEX ARRAY THAT WILL BE USED FOR DFT.
        DO 610 J=1,ISIZE
        B(J)=IDSP(J+J1)
        WNDO = 1.0       ;Defaults to rectangular window
        IF(IHAM .EQ. 1) WNDO=0.54-0.46*COS(2.0*3.1415926536*
     /  (J-1)/ISIZE)             ;Equation for Hamming window
        B(J) = WNDO * B(J)
        DTARAY(J)=B(J)
 610    CONTINUE
        ;DO DFT, LARAY=# OF POINTS, INVER SPECIFIES FORWARD TRANSFORM
        LARAY=ISIZE
        INVER=0
        CALL DFT5(DTARAY,LARAY,INVER)
        ;FIND MAGNITUDE OF DFT RESULTS TO GET FREQ. COMPONENTS.
        DO 625 I=1,ISIZEH
        B(I)=CABS(DTARAY(I)/LARAY)
        IF(IHAM .EQ. 1) B(I)=B(I)/0.54
 625    CONTINUE
C
C PREEMPHASIZE HIGH FREQUENCY COMPONENTS.
C
        IF (IPRE.NE.1) GO TO 45
        ;FIND 1ST COMPONENT TO START PREEMPHASIS AT.
        IFREQ1=IFIX(FREQ/SFREQ*ISIZE)+1
        DO 700 I=IFREQ1,ISIZEH
        ;CONVERT INTEGER VARIABLES TO REALS.
        RI=I
        RIFREQ1=IFREQ1
        RIDB=IDB
        ;MODIFY MAGNITUDE OF COMPONENTS.
        B(I)=B(I)*(10**(RIDB/20*ALOG10(RI/RIFREQ1)/ALOG10(2.0)))
 700    CONTINUE
C
C FREQUENCY VECTOR COMPRESSED TO 16 CHANNELS IF COMPRESSION WAS SELECTED.
C
 45     IF (IPRESS.NE.1) GO TO 30
        ;FIND NUMBER OF COMPONENTS TO COMBINE INTO 1 COMPONENT
        JNUM=ISIZEH/16
        JJ1=1
        JJ2=JNUM
        DO 20 I=1,16
        SUM=0.0
        DO 10 J=JJ1,JJ2
 10     SUM=SUM+B(J)
        B(I)=SUM
        JJ1=JJ1+JNUM
        JJ2=JJ2+JNUM
 20     CONTINUE
C
```

```
C NORMALIZATION OF ENERGY IN VECTORS.
C
 30     IF (IPRESS.EQ.1) ISIZEH=16
        SUME=0.0
        DO 33 J=1,ISIZEH
        SUME=SUME+(B(J)**2)
 33     CONTINUE
        ENERGY=SQRT(SUME)
        WRITE(4) ISAMPP,ENERGY  ;SAVE INDIVIDUAL ENERGYS OF VECTORS.
        ISAMPP=ISAMP+1
        IF (ENERGY.GT.ENTHRES) GO TO 32
        ;INCREASE VALUE OF ENERGY TO ATTENUATE COMPONENTS
        ENERGY=5*ENTHRES
 32     CONTINUE
        IMULT=32767 ;RANGE OF CONPONENTS AFTER NORMALIZATION.
        DO 34 J=1,ISIZEH
        B(J)=(B(J)/ENERGY)*IMULT
        ; SAVE TOTAL ENERGY FOR EACH COMPONENT FOR THIS UTTERANCE.
        RENERGY(J)=RENERGY(J)+B(J) ;SUM UP TOTAL ENERGY
 34     CONTINUE
C
C ARRAY VALUES CONVERTED TO INTEGER FORM
C
        DO 240 JJ=1,ISIZEH
        IBI(JJ)=IFIX(B(JJ))
 240    CONTINUE
        ;
        ;SAVE VECTOR COMPONENTS FOR USE LATER BY 'PPGEN'
        ;
        CALL WRITR(2,ISAMP,IBI,1,IER)
        IF(IER.NE.1) TYPE "ERROR ON WRITR,IER=",IER
        ;
        ;IF SGRAM IS NOT DESIRED,SKIP OVER SECTION TO SCALE THE COMPONENTS
        ;TO MAX. VALUES OF 10 AND THEN OUTPUT THE SGRAM CHARACTERS.
        ;
        IF (ISPEC.NE.1) GOTO 330
        DO 245 I=1,ISIZEH
        IBI(I)=IFIX(FLOAT(IBI(I))/32767*10)+1
        ;MAKE SURE COMPONENT VALUES ARE IN CORRECT RANGE (1-10).
        IF (IBI(I).LE.0) IBI(I)=1
        IF(IBI(I).GT.10) IBI(I)=10
 245    CONTINUE
C
C DETERMINE AND OUTPUT THE CHARACTERS THAT WILL CONSTRUCT THE SPECTROGRAM
C
        DO 320 I=1,IWIDTH  ;REPEAT SYMBOLS IWIDTH NUMBER OF TIMES.
        CALL BYTEOUT(ICHAN,IPLOT)  ;SEND PLOT COMMAND TO PRINTRONIX
        ;SEND 1ST DOT ROW FOR ALL THE COMPONENTS
        DO 250 JJ=1,ISIZEH
        JS=IBI(JJ)
        DO 250 J=1,IHEIGHT
        CALL BYTEOUT(ICHAN,ISYMBOL1(JS))
 250    CONTINUE
        CALL BYTEOUT(ICHAN,ILF)  ;SEND LINE FEED TO TERMINATE LINE
```

```
      CALL BYTEOUT(ICHAN,IPLOT)
      ;SEND 2ND DOT ROW FOR ALL THE COMPONENTS
      DO 260 JJ=1,ISIZEH
      JS=IBI(JJ)
      DO 260 J=1,IHEIGHT
      CALL BYTEOUT(ICHAN,ISYMBOL2(JS))
260   CONTINUE
      CALL BYTEOUT(ICHAN,ILF)
      CALL BYTEOUT(ICHAN,IPLOT)
      ;SEND 3RD DOT ROW FOR ALL THE COMPONENTS
      DO 270 JJ=1,ISIZEH
      JS=IBI(JJ)
      DO 270 J=1,IHEIGHT
      CALL BYTEOUT(ICHAN,ISYMBOL3(JS))
270   CONTINUE
      CALL BYTEOUT(ICHAN,ILF)
      CALL BYTEOUT(ICHAN,IPLOT)
      ;SEND 4TH DOT ROW FOR ALL THE COMPONENTS
      DO 280 JJ=1,ISIZEH
      JS=IBI(JJ)
      DO 280 J=1,IHEIGHT
      CALL BYTEOUT(ICHAN,ISYMBOL4(JS))
280   CONTINUE
      CALL BYTEOUT(ICHAN,ILF)
      CALL BYTEOUT(ICHAN,IPLOT)
      ;SEND 5TH DOT ROW FOR ALL THE COMPONENTS
      DO 290 JJ=1,ISIZEH
      JS=IBI(JJ)
      DO 290 J=1,IHEIGHT
      CALL BYTEOUT(ICHAN,ISYMBOL5(JS))
290   CONTINUE
      CALL BYTEOUT(ICHAN,ILF)
      CALL BYTEOUT(ICHAN,IPLOT)
      ;SEND 6TH DOT ROW FOR ALL THE COMPONENTS
      DO 300 JJ=1,ISIZEH
      JS=IBI(JJ)
      DO 300 J=1,IHEIGHT
      CALL BYTEOUT(ICHAN,ISYMBOL6(JS))
300   CONTINUE
      ;PRINT A DASH AFTER EVERY 10 VECTORS
      IF (ICOUNT.NE.10) GOTO 310
      CALL BYTEOUT(ICHAN,IDASH)
      ICOUNT=0
310   CALL BYTEOUT(ICHAN,ILF)
320   CONTINUE
      CALL BYTEOUT(ICHAN,IBLANK);SEND A BLANK TO INSURE LAST CHARACTER SENT
330   ICOUNT=ICOUNT+1
      ;RESTORE ISIZEH INCASE IT WAS CHANGED FOR COMPRESSION
      ISIZEH=ISIZE/2
C
C END OF SCRAM CONSTRUCTION
C
      ISAMP=ISAMP+1
      ;CHECK TO SEE IF ANOTHER VECTOR CAN BE FORMED FROM THIS BLOCK OF SPEECH
```

```
      J1=J1+ISIZE
      IF (J1.LT.256) GO TO 600
1000  CONTINUE
      ;
      ;END OF MAIN LOOP OF PROGRAM
      ;
      DO 1010 I=1,ISIZEH
1010  WRITE(1) I,RENERGY(I) ;SAVE TOTAL ENERGY ARRAY
      CALL RESET
      WRITE(10)"<7>","<7>","<7>"  ;RING BELL TO WAKE UP OPERATOR
      STOP
      END
```

PROGRAM TEKTALK

File: TEKTALK

Language: Fortran 5

Date: October 20, 1982

Author: K. Seelandt, modified by J. Fletcher

Subject: Digitized speech output

Calling Sequence: TEKTALK

| ARGUMENT | TYPE | PURPOSE |
|---|---|---|
| SFREQ | Real | Value set for the sampling frequency |
| IDB | Integer | Number of DB to preemphasize |
| FREQ | Real | Starting frequency of pre-emphasize |
| ISIZE | Integer | DFT sampling size |
| ENTHRES | Real | energy threshold for normal-ization |
| PARAM (1) & FILE NAM | String | name of desired speechfile |
| TITLE | String | Title for spectrogram |
| ISTART | Integer | First block in speechfile to be used in spectrogram |
| ILAST | Integer | Last block in speechfile to be used in spectrogram |
| DTARAY | Real Array | Holds frequency components to be used in DFT |
| RENERGY | Real Array | Holds energy for each spectral component in the sentence |
| B | Real Array | Holds windowed frequency com-ponents |

| | | |
|---|---|---|
| IBI | Integer | Holds integer values of frequency components after DFT is performed |
| IDSP | Integer Array | Holds a block of components read from speechfile |
| ICONTROL | Integer Array | Holds screen command for TEKtronix 1040 terminal |
| IH | Integer Array | Holds position of cross hairs |
| IL | Integer Array | Holds position of cross hairs |
| IMASK | Integer Array | Holds constants used for ANDing and ORing |
| PARAM (8) | Real | Starting position to heard in speechfile |
| PARAM (9) | Real | Ending position to heard in speechfile |
| PARAM (10) | Real | Number of times segement to be heard |

PURPOSE:

This program is used to plot the spectrogram and allow the user to listen to any part of the speechfile he wants, as many times as he wants.

DESCRIPTION

Location: FLETCHER. DR

ARGUMENT STRUCTURE:

| | | |
|---|---|---|
| PARAM (1)& FILENAM | = | name of speechfile in 13 character Hollerith string format |
| TITLE | = | title for the spectrogram in 79 character Hollerith string format |
| ISTART | = | starting block of speechfile to be shown using spectrogram |
| ILAST | = | last block of speechfile for spectrogram; total number of blocks must not exceed 20 |
| IDB | = | 6dB was used exclusively in this thesis |

FREQ = 500.0 was used exclusively

ISIZE = number of samples for each time slice, 64 was used exclusively

ENTHRES = threshold for energy the value 10 was used

PROGRAM USE:

This program can be used only on the Tedtronix 4010 terminal using the Data General A/D/A converter. The A/D/A converter must be set up prior to the running of this program as per instructions in the thesis text. The speech output is through the Rockland filter and Crown amplifier. The user can select any segement of speech he wants to listen to by displaying up to 20 blocks of speech on the terminal and using the cross hairs to delimit the segment of speech he wants to hear and the number of times he wants it repeated. This program is especially useful for listening to speech segments to pick phoneme prototypes. To load this program, use the following command:

```
RLDR TEKTALK BYTEPAC TEKDOT TEKLINE TEKTONE
EDFT.LB @FLIB@
```

Also, make sure TALK.SV is in the same directory since this is swapped with TEKTALK to output the speech.

FILES CREATED:

TENERGY: contains component number and the total energy in the speechfile for each spectral component

IENERGY: contains the vector number and the total energy for each vector

DCOMPS: contains the DFT frequency components of the vectors; each vector is stored as a record

PARAM.AD: contains file name of speech, starting and ending work of segment of speech to be heard and number of times to repeat it; data written sequentially in the file using WRSEQ

```
C       ##########################################################
C       ##########################################################
C       **                                                      **
C       **      PROGRAM:   T E K T A L K                        **
C       **                                                      **
C       **               WRITTEN BY:  LT KARL SEELANDT          **
C       **                                                      **
        **               REVISED BY:  CAPT JAMES E. FLETCHER    **
        **                                                      **
C       **                      DATE: 12 OCT. '82               **
C       **                                                      **
C       ##########################################################
C       ##########################################################
C
C       Load TEKTONE, TEKDOT, TEKLINE, BYTEPAC and EDFT.LB
C       along with this program using the RLDR command.
C
C               NOTE:  THIS ROUTINE IS FOR FORTRAN 5 !!
C
C******     The spectrogram portion of this program was derived from
C       a program written by Lt. Mark Felkey while working on his
C       thesis here at AFIT.
C
C******     This program will accept as an input, a file containing
C       from 1 to 88 blocks of digitized speech.  The output will
C       be a spectrogram of a portion of the file, with a length up to
C       80 vectors, or 0.64 seconds of speech (with a 64 sample DFT).
C       Again if the 64 sample DFT is used the frequency spectrum of
C       the spectrogram consists of 32 separate levels from DC to 4KHz.
C       When this program uses the 64 sample DFT the vectors have a
C       length of 8msec since the sampling rate of the analog speech
C       signal is 8KHz.  The maximum length of a speechfile is
C       currently 88 blocks / 22528 words, this equates to a maximum
C       spectrogram length of 352 vectors ( = 22528 / 64 ).
C
C******     After taking the DFT of the digitized speech the high
C       frequency components of each time vector are preemphasized,
C       the components of the vectors are then energy thresholded
C       and finally they are energy normalized on a scale
C       of 0 thru 9 for gray-scale encoding.  After all this has
C       been done the spectrogram is displayed on the TeKtronix
C       4010 Graphics Terminal.  Once the spectrogram is completed
C       the terminal is left in the GIN-mode and will display the
C       cursors.  The cursors can be adjusted by the operator to
C       input two X-values from the screen to the computer, they
C       determine what portion of the spectrogram will be heard when
C       using the audio output program.  This allows the operator to
C       listen to any segment of speech displayed on the spectrogram
C       enabling him/her to determine an optimal set of prototype
C       phonemes to be used for the recognition of phonemes in
C       connected speech.
C
C
C******     NOTE:  In addition to the above listed information
```

```
C       this routine will also allow the user to choose the
C       of windowing used in creating the spectrogram.  As the
C       operator requests, the window will be a rectangular non-
C       overlapping one or a Hamming non-overlappingone.
C       The window multiplier for a rectangular window is 1.
C       The window multiplier used for the Hamming window is
C       the same as that mentioned by Oppenhiem & Shafer in
C       Digital Signal Processing, on page 242.............
C               Also note that when using other than a
C       rectangular window the frequency components should
C       be multiplied be some multiplication factor in
C       order to keep the relative magnitudes the same.
C       For a Hamming window this factor is 1/0.54 and is
C       done right after the call to the DFT routine.
C
C
C
C******     The number of times a segment of speech is heard can be
C       operator adjusted from the Tektronix terminal by the keys
C       hit to send the cursor information about the final position
C       of the two positions sent to hear a segment.  For example
C       hitting a 1 then 9 will allow the user to hear the speech
C       segment 19 times, then the cursors are displayed again and
C       the process can be continued.
C
C
C******     Set up variables to be used.
C
        DIMENSION IH(2),IL(2),IWORD1(2),IWORD2(2),I(3),IMASK(3)
        DIMENSION DTARAY(256),IDATE(3),B(256),IBI(128),IDSP(256)
        DIMENSION ICONTROL(7),RENERGY(64),ISTORE(13)
        INTEGER FILENAM(13),TITLE(79),PARAM(10)
C
        DATA IMASK/37K,17400K,17K/
        DATA ICONTROL/033K,014K,032K,005K,035K,037K,007K/
C
C** ASCII equivalent= ESC  FF   SUB  ENQ   GS   US   BEL.
C
        COMPLEX DTARAY
C
C******     Define critical variables used in the program, some
C       variables can be user-defined by optionally compiling
C       statements preceeded by an 'X' using the FORT/X command.
C
  5     IFLAG=0                         ;what job is next.
        ISAMP=0                         ;counts vectors created.
        IPRE=1                          ;signal to preemphasize.
        IDB=6                           ;DB's to preemphasize by.
        SFREQ=8000.0                    ;sampling frequency
        FREQ=500.0                      ;start preemphasis here.
        ISIZE=64                        ;DFT sample size.
        IHAM=0                          ;what type of windowing
                                ;will be done default = rectang.
C
```

```
      IBYTE=999                             ;init. output routine.
      CALL BYTEPAC(IBYTE)      ;
    6 CALL RESET
C
C******    Input control variables and print a heading.
C
      OPEN 1,"TENERGY"                      ;contains total energy of
                                            ;speechfile at each freq.
      OPEN 4,"IENERGY"                      ;contains energy of
                                            ;each individual vector.
C
C******    Make available some basic instructions for the user and
C     input the name of the speechfile to be used.
C
C
      TYPE"<15> <15> "
      TYPE"    W E L C O M E   T O   T E K T A L K "
      TYPE"    *************   ***   ************* <15>"
      TYPE"               developed by:     "
      TYPE"            LT KARL G. SEELANDT      "
      TYPE"               revised by:  "
      TYPE"          CAPT JAMES E. FLETCHER <15><15> "
      ACCEPT"DO YOU WANT TO CONTINUE/GET MORE INFO
     / ; ENTER 1/0 ",ICONT
C
      IF(ICONT.EQ.1)GO TO 8
      CALL BYTEPAC(ICONTROL(1))
      CALL BYTEPAC(ICONTROL(2))
      CALL FDELAY(10)
C
      TYPE"        This routine was developed as an"
      TYPE"   interactive routine to be used as a tool"
      TYPE"   that would enable a user to work in the"
      TYPE"   area of prototype phoneme developement."
      TYPE"   TEKTALK allows a user to simultaneously"
      TYPE"   see a picture of speech (a spectrogram)"
      TYPE"   while hearing different segments of the"
      TYPE"   same. <15> <15>          "
      TYPE"        For this program to work make sure"
      TYPE"   the following rules are followed: "
      TYPE"   1....TALK.SV has been compiled and loaded"
      TYPE"        in your directory.  "
      TYPE"   2....Make sure DP4F:SAM has been initialized"
      TYPE"   3....The proper hardware connections have"
      TYPE"        been made to the A/D/A converter."
      TYPE"   4....Make sure the cables have been switched"
      TYPE"        in the back of the Crown amplifier and"
      TYPE"        Rockland Hi/Lo pass filter."
      TYPE"   5....FILE SPACE must be available, about"
      TYPE"        100 to 200 blocks.  "
      ACCEPT"TO CONTINUE, HIT RETURN."
      READ(11,2) IHOLD
    2 FORMAT(S1)
      CALL BYTEPAC(ICONTROL(1)) ;   Clear screen and home char
```

```
      CALL BYTEPAC(ICONTROL(2)) ;   pointer to prevent overwriting.
      CALL FDELAY(10)
  8   ACCEPT" ENTER NAME OF SPEECHFILE: "
      READ(11,10)PARAM(1)
 10   FORMAT(S13)
         DO 1 J=1,7
         FILENAM(J) = PARAM(J)
  1   CONTINUE
C
C
101   ACCEPT"ENTER TITLE FOR SGRAM (79 characters or less):(15)"
      READ(11,11)TITLE(1)
 11   FORMAT(S79)
C
      CALL OPEN(5,FILENAM,1,512,IER1) ;channel to speechfile
      IF(IER1.NE.1)TYPE"ERROR ON OPEN, IER1= ",IER1
C
C
C
 19   TYPE"NOTE: You must do a spectrogram of 20 blocks"
      TYPE"      or less (80 or less vectors), so that"
      TYPE"      it will fit on the Tektronix screen !!"
      TYPE"      This equals 0.64 seconds of speech, max."
      TYPE"   "
 20   ACCEPT"ENTER FIRST BLOCK TO BE READ :  ",ISTART
      ACCEPT"ENTER LAST BLOCK TO BE READ  :  ",ILAST
      TYPE"WHAT TYPE OF WINDOW IS DESIRED"
      ACCEPT"HAMMING / RECTANGULAR (1 / 0): ",IHAM
C
C
X     ACCEPT"PREEMPHASIZE HIGH FREQ.? (1=YES/ 2=NO): ",IPRE
X     IF(IPRE.EQ.1)ACCEPT"ENTER DB's per OCTAVE : ",IDB
X     IF(IPRE.EQ.1)ACCEPT"ENTER STARTING FREQ.  : ",FREQ
X     ACCEPT"ENTER DFT SAMPLE SIZE (= samples/vector) :",ISIZE
C
C
      ISIZEH=ISIZE/2                   ;number of distinct freq.
C
C
      OPEN 19,"DCOMPS",LEN=ISIZE       ;store freq components.
C
 30   ACCEPT"ENTER ENERGY THRESHOLD :        ",ENTHRES
C
      TYPE" (15) "
      TYPE"           ****** N O T I C E ******  (15) "
      TYPE"      While in the interactive mode, EACH time"
      TYPE"   you listen to a portion of a spectrogram the   "
      TYPE"   following information is written to a file:(15)"
      TYPE"   1.      Date and time.           "
      TYPE"   2.      Name of speechfile.      "
      TYPE"   3.      Spectrogram heading.     "
      TYPE"   4.      And for each segment.    "
      TYPE"     a.    A numbering.             "
      TYPE"     b.    Position of the first    "
```

```
      TYPE"          vector heard (in words)."
      TYPE"     c.   Number of words heard.  "
      TYPE"     d.   Segment length (sec). (15)"
      ACCEPT"TO CONTINUE, HIT RETURN"
      READ(11,2) IHOLD1
      CALL BYTEPAC(ICONTROL(1))
      CALL BYTEPAC(ICONTROL(2))
      CALL FDELAY(10)
C
C
      TYPE"(15)  (15) "
      TYPE" WHILE IN THE INTERACTIVE MODE: "
      TYPE"  "
      TYPE" Perform following steps on Tektronix terminal:"
      TYPE" (15) "
      TYPE"1.  Position crosshairs at FRONT of first speech "
      TYPE" vector you want to hear."
      TYPE"2.  Hit twice any key (on Tektronix) to send cursor"
      TYPE" position to the computer."
      TYPE"3.  The cursor is displayed after a beep, and you "
      TYPE" should now position crosshairs BEHIND the last "
      TYPE" vector you want to hear."
      TYPE"4.  Next hit the keys that represent the number"
      TYPE" of times you want to hear the section of speech"
      TYPE" just marked off, from 1 thru 99 times.  "
      TYPE" For example, entering a 1 then a 5 will let"
      TYPE" you hear the segment 15 times. (15) "
      TYPE"***To STOP send same crosshair position twice. "
      TYPE"(15) (15) "
      ACCEPT"TO CONTINUE, HIT RETURN."
      READ(11,2) IHOLD2
C******    Print a heading on the Tektronix screen.
C
C
C******    For two character commands must send both in
C     one word, so use BYTEPAC to pack two bytes into
C     one memory word.  Can just use a WRITE BINARY
C     for one character (ASCII) commands.
C
 31   CALL BYTEPAC(ICONTROL(1))      ;send ESC FF
      CALL BYTEPAC(ICONTROL(2))      ;to blank screen
C
      CALL DATE(IDATE,IER3)          ;get the date.
      IF(IER3.NE.1)TYPE"ERROR ON DATE, IER3= ",IER3
C
      CALL FGTIME(IHOUR,IMIN,ISEC)   ;get the time.
C
C******    To output heading send position that will be the
C     lower left corner of first character, then go back to
C     the Alpha mode and write this portion of the heading.
C
      CALL FDELAY(20)                ;time delay to allow
                                     ;screen to clear.
      CALL TEXDOT(740,200)           ;start of line.
```

```
      WRITE BINARY (10) ICONTROL(6)  ;to get to Alpha.
      WRITE(10,100)TITLE(1)           ;send title
  100 FORMAT(S79)                     ;
C
      CALL TEKDOT(710,200)            ;start of next line
      WRITE BINARY (10) ICONTROL(6)  ;go to Alpha mode
      WRITE(10,110)FILENAM(1)         ;send filename
  110 FORMAT("THIS IS A SPECTROGRAM OF: ",S13)
C
      CALL TEKDOT(680,200)            ;start of next line
      WRITE BINARY (10) ICONTROL(6)  ;go to Alpha mode.
      WRITE(10,112)ISTART,ILAST       ;send block numbers
  112 FORMAT("FIRST BLOCK= ",I2,5X,"LAST BLOCK= ",I2)
C
      CALL TEKDOT(650,200)            ;start of next line
      WRITE BINARY (10) ICONTROL(6)  ;go to Alpha mode.
      WRITE(10,120)IDATE                      ;send date
  120 FORMAT("DATE: ",2I3,I5)         ;
C
      CALL TEKDOT(650,478)            ;start of next output.
      WRITE BINARY (10) ICONTROL(6)  ;go to Alpha mode.
      WRITE(10,125)IHOUR,IMIN,ISEC    ;send time
  125 FORMAT("TIME: ",3I3)            ;
C
      CALL TEKDOT(620,200)            ;start of next line.
      WRITE BINARY (10) ICONTROL(6)  ;go to Alpha mode.
      WRITE(10,130)IDB                ;send preemphasis info.
  130 FORMAT("FREQ. PREEMPHASIS, DB/OCTAVE: ",I3)
C
      CALL TEKDOT(590,200)            ;start of next line.
      WRITE BINARY (10) ICONTROL(6)  ;go to Alpha mode.
      WRITE(10,140)FREQ               ;start of preemphasis.
  140 FORMAT("PREEMPHASIS STARTS AT, FREQ.: ",F5.0)
C
      CALL TEKDOT(560,200)            ;start of next line.
      WRITE BINARY (10) ICONTROL(6)  ;go to Alpha mode.
      WRITE(10,150)ENTHRES            ;send threshold value.
 150  FORMAT("RMS ENERGY THRESHOLD: ",F6.1)
C
      CALL TEKDOT(530,200)            ;start of next line.
      WRITE BINARY (10) ICONTROL(6)  ;go to Alpha mode.
      WRITE(10,160)ISIZE              ;send DFT size.
  160 FORMAT("DFT SAMPLE SIZE: ",I3)
C
C
      CALL TEKDOT (500,200)           ;start next line.
      WRITE BINARY (10) ICONTROL(6)   ;Alpha
      IF(IHAM.EQ.1)WRITE(10,161)      ;Hamming
      IF(IHAM.NE.1)WRITE (10,162)     ;Rectangular.
 161  FORMAT("HAMMING WINDOW USED")
 162  FORMAT("RECTANGULAR WINDOW USED")
C
C
C
```

```
C******     Mark off the time axis, in vectors.
C
        DO 165 JJ=1,5
           J=JJ-1
           INUM=(ISTART + J*5)*4                    ;vector marking.
           IX=9 + J*240                  ;marking position.
           CALL TEKDOT(40,IX)            ;start line.
           WRITE BINARY (10) ICONTROL(6)  ;to Alpha mode.
           WRITE(10,164)INUM             ;send marking.
  164      FORMAT(I3)                    ;
  165   CONTINUE
C
C******     Write out an axis label.
C
        CALL TEKDOT(10,419)              ;start line.
        WRITE BINARY (10) ICONTROL(6) ;to Alpha mode.
        WRITE(10,168)            ;send label.
  168   FORMAT("VECTOR NUMBER")
C
C
C
C******     Print an axis for the spectrogram on the screen
C       of the Tektronix, then delineate the X-axis every
C       vector and delineate the Y-axis every frequency
C       component.  The subroutine TEKLINE(IY,IX,IY1,IX1)
C       draws a line from the point (IX,IY) to (IX1,IY1).
C
        CALL TEKLINE(474,991,89,991)             ;Y-axis, right.
        CALL TEKLINE(474,29,89,29)               ;Y-axis, left
        CALL TEKLINE(89,29,89,990)               ;X-axis done
C
        DO 170 IXD=30,990,12                     ;delineate the
           CALL TEKLINE(88,IXD,78,IXD)           ;X-axis first.
  170   CONTINUE
C
        DO 175 IXD=30,990,120                    ;put extra mark
           CALL TEKLINE(78,IXD,70,IXD)           ;each 10 vectors.
  175   CONTINUE
C
        DO 180 IYD=90,474,12                     ;delineate the
           CALL TEKLINE(IYD,28,IYD,18)           ;Y-axis, left.
  180   CONTINUE
C
        DO 185 IYD=90,474,96                     ;put extra mark
           CALL TEKLINE(IYD,18,IYD,8)            ;every 1KHz.
  185   CONTINUE
C
        DO 190 IYD=90,474,12                     ;delineate the
           CALL TEKLINE(IYD,992,IYD,1002)               ;Y-axis, right.
  190   CONTINUE
C
        DO 195 IYD=90,474,96                     ;put extra mark
           CALL TEKLINE(IYD,1002,IYD,1012)              ;every 1 KHz.
  195   CONTINUE
```

```
C
C
C
C######     Now create the spectrogram in the steps outlined:
C
C       1.      Clear array that is used to add up energy
C               in a sentence.
C
C       Main loop of program
C
C       2.      Read a block of speech.
C       3.      Set B(J) = magnitude of speech components.
C               Then window as requested!
C       4.      Then set DTARAY = B(J), so array is complex.
C       5.      Call DFT routine to get freq. info.
C
C       Sub-loop of program (once for each freq)
C
C       a.      Get magnitude of freq. components.
C               If Hamming window used normalize the
C               magnitude of the freq. components
C               wrt a rectangular window (=1).
C       b.      Preemphasize frequencies above 500 Hz.
C       c.      Save energy components in IENERGY.
C       d.      Energy normalize the vector.
C
C       End of sub-loop
C
C       6.      Threshold vector wrt. energy.
C       7.      Sum total energy @ each freq in TENERGY.
C       8.      Change B(J)'s to integers.
C       8.      Store freq components of each vector
C               in DCOMPS.
C       9.      Scale freq components from 1 to 10.
C       10.     Output to screen one vector of info.
C       11.     Change vector number and GO TO #1.
C       12.     When done save TENERGY and RING BELL.
C
C
C######     Now do it to it.
C
        DO 200 I1=1,ISIZEH              ;clear file used to
  200      RENERGY(I1)=0.0                      ;sum up energy.
C
C
        DO 900 I2=ISTART,ILAST          ;begin main loop
           CALL RDBLK(5,I2,IDSP,1,IER)  ;input speech info.
           IF(IER.NE.1)TYPE"ERROR ON RDBLK = ",IER
           J1=0                    ;Keeps track of whether to read
                                   ;another block of speech.
C
           WNDO=1.0                ;Default to rectang. window.
  300      DO 310 J=1,ISIZE             ;form complex array
              B(J)=IDSP(J+J1)                   ;J1 = block number.
```

```
C
            IF (IHAM.EQ.1) WNDO = 0.54 - (0.46 * COS( 2. *
     /      3.1415926536 * (J - 1)/ISIZE))
C
            B(J) = WNDO * B(J)
C
C
            DTARAY(J)=B(J)                        ;must call DFT with
                                        ;complex array.
  310    CONTINUE
C
         LARAY=ISIZE                    ;size for DFT call.
         INVER=0                                  ;indicate DFT direction.
         CALL DFT5(DTARAY,LARAY,INVER)            ;get DFT for this vector
C
         DO 325 I3=1,ISIZEH             ;get freq component magn.
            B(I3)=CABS(DTARAY(I3)/LARAY)          ;
C
            IF (IHAM.EQ.1) B(I3) = B(I3) / 0.54           ;normalize
                        ;to equivalent magnitude as a
                        ;rectangular window.
C
  325    CONTINUE
C
         IF(IPRE.NE.1)GO TO 400                   ;to not preemphasize
         IFREQ1=IFIX(FREQ/SFREQ*ISIZE)+1          ;first freq preemph.
C
         DO 400 IN=IFREQ1,ISIZEH                  ;preemph. freq components.
            RI=FLOAT(IN)                          ;change integer variables
         RIFREQ1=FLOAT(IFREQ1)                    ;into real variables.
            RIDB=FLOAT(IDB)                       ;
      ;Perform freq preemphasis next.
            B(IN)=B(IN)*(10.**(RIDB/20*ALOG10(RI/RIFREQ1)/ALOG10(2.0)))
  400    CONTINUE
C
         SUME=0.0                       ;energy normalize vectors.
         DO 432 J=1,ISIZEH              ;sum=squareroot of the
            SUME=SUME+(B(J)**2)                   ;sum of the squares.
  432    CONTINUE
         ENERGY=SQRT(SUME)              ;
C
         ISAMPP=ISAMP+1                           ;vector counter.
         WRITE(4) ISAMPP,ENERGY                   ;store energy.
C
         IF(ENERGY.GT.ENTHRES)GO TO 433           ;if below threshold
      ;increase energy to attenuate components.
         ENERGY=5.*ENTHRES              ;
  433    CONTINUE
         IMULT=32767            ;max possible energy value.
         DO 434 J=1,ISIZEH      ;change to integers.
            B(J)=(B(J)/ENERGY)*IMULT  ;normalize.
C
            RENERGY(J)=RENERGY(J)+B(J)            ;sum energy.
  434    CONTINUE
```

```
C
            DO 500 JJ=1,ISIZEH          ;array values converted
               IBI(JJ)=IFIX(B(JJ))              ;to integer format.
  500       CONTINUE
C
         ;save vector components for use later
            CALL WRITRW(19,ISAMP,IBI,1,IER4)
            IF(IER4.EQ.1.OR.IER4.EQ.3121)GO TO 510       ;
            TYPE" WRITRW ERROR, IER4= ",IER4         ;
C
  510       DO 545 IN=1,ISIZEH          ;scale components
               IBI(IN)=IFIX(FLOAT(IBI(IN))/32767*10)+1
               IF(IBI(IN).LE.0)IBI(IN)=1        ;from 1 to 10.
               IF(IBI(IN).GT.10)IBI(IN)=10
  545       CONTINUE
C
C
C******      Output the components of this vector now.
            DO 700 IN=1,ISIZEH          ;for each freq. component
               CALL TEKTONE(ISAMP,IN,IBI(IN))    ;draw pixel
  700       CONTINUE
C
C
            ISAMP=ISAMP+1                       ;go to next vector
            J1=J1+ISIZE          ;can another vector be formed
            IF(J1.LT.256)GO TO 300              ;from this block.
C
  900    CONTINUE
C
C******       End of main program loop.
C
         DO 910 IN=1,ISIZEH             ;save total energy
 910        WRITE(1) I,RENERGY(IN)              ;
C
C
         TYPE" <7> <7> <7> <7> "        ;make up operator
                                        ;for interactive mode.
C******       Now go into interactive mode on the Tektronix
C        by making use of the Gin mode of the tektronix.
C        This section of the program will do the following:
C
C                a.      Send ESC-SUB to put the Tektronix
C                        into Gin mode (must use the Bytepac
C                        routine so both bytes are sent in a
C                        single word).
C                b.      Tell user what to do now.
C                c.      Decode first group of bytes sent
C                        from the Tektronix into X,Y tekpoint
C                        position of starting position.
C                d.      Decode second group of bytes sent
C                        into the number of times you want
C                        to hear portion of spectrogram,
C                        and the final X,Y tekpoint position.
C                e.      Check to be sure that the second
```

```
C                       position is not equal to the first
C                       position (which occurs when user
C                       wants to leave the program).
C               f.      Transfer control to the program
C                       which is used to I/O speech, this is
C                       program TALK.SV which is used to
C                       output user specified segments
C                       of a speechfile.  This is done utilizing
C                       the eclipse's D/A/D converter.
C               g.      Continue with the use of the
C                       cursor routine.
C
C       NOTE:   The following describes the bytes sent out
C               from the Tektronix terminal when two Keys
C               are hit while in the Gin mode.
C
C       +-------+-------+-------+-------+-------+-------+
C        byte 1  byte 2  byte 3  byte 4  byte 5  byte 6
C       +-------+-------+-------+-------+-------+-------+
C  first byte sent                              last byte sent
C
C       byte 1.......contains the ASCII code first key hit
C       byte 2.......contains TeKtronix coded Hi X 5-bit byte
C       byte 3.......contains TeKtronix coded Lo X 5-bit byte
C       byte 4.......contains TeKtronix coded Hi Y 5-bit byte
C       byte 5.......contains TeKtronix coded Lo Y 5-bit byte
C       byte 6.......contains the ASCII code last key hit
C
C
C
        CALL RESET
C
C******    Write heading to file ISTORE.
C
        CALL FOPEN(3,"ISTORE")                  ;
C
        CALL FGTIME(IHOUR,IMIN,ISEC)            ;Current time.
C
        WRITE(3,917)                            ;
 917    FORMAT(12X,"DOCUMENTATION OF WORK FROM");
        WRITE(3,918)                            ;
 918    FORMAT(19X,"T E K T A L K ")            ;
        WRITE(3,920)IDATE                       ;
  920   FORMAT(10X,"DATE : ",2I3,I5)            ;
        WRITE(3,921)IHOUR,IMIN,ISEC             ;
  921   FORMAT(10X,"TIME : ",3I3)                       ;
        WRITE(3,922)FILENAM(1)                  ;
  922   FORMAT(10X,"NAME OF SPEECHFILE : ",S13) ;
        WRITE(3,923)TITLE(1)                    ;
  923   FORMAT(" ",S79)                         ;
        ICOUNT=1                ;counter for number of segments.
C
C
C
```

```
      IT=0
 990  CALL RESET
      CALL OPEN(3,"ISTORE",0,IERA)    ;documentation file.
      IF(IERA.NE.1)TYPE"ERROR ON OPEN, IERA: ",IERA
C
C
C
 1000 CALL BYTEPAC(ICONTROL(1))              ;send ESC-SUB for
      CALL BYTEPAC(ICONTROL(3))              ;for Gin mode.
C
      READ BINARY (11) I(1),I(2),I(3)
C
      IF(IT.EQ.0)GO TO 1010                  ;first position
      IHEAR1=IAND(IMASK(2),I(1))             ;# of times you
      IHEAR1=ISHFT(I(1),-8)                  ;hear portion.
C
      IHEAR2=IAND(IMASK(1),I(3))             ;2nd key hit
C
 1010 IH(1)=IAND(IMASK(1),I(1))              ;decode the
      IH(1)=ISHFT(IH(1),5)                   ;X position
      IL(1)=IAND(IMASK(2),I(2))              ;of cursor
      IL(1)=ISHFT(IL(1),-8)                  ;
C
      IH(2)=IAND(IMASK(1),I(2))              ;decode the
      IH(2)=ISHFT(IH(2),5)                   ;Y position
      IL(2)=IAND(IMASK(2),I(3))              ;of cursor.
      IL(2)=ISHFT(I(3),-8)                   ;
C
C******     The first and last bytes from the Tektronix carry
C     only the ASCII code for the first and second keys hit
C     at the Tektronix terminal.  Hitting the first key causes
C     cursor position information to be sent and the second
C     key is hit to fill up the last word (16 bit) of infor-
C     mation sent to the computer.  These two bytes are
C     examined and decoded upon receipt of the final cursor
C     position, enabling the user to specify the number of
C     times a section of speech will be heard.
C
      IF(IT.EQ.1)GO TO 1100
C
      DO 1050 IN=1,2                         ;decode position.
         IWORD1(IN)=IOR(IH(IN),IL(IN))
 1050 CONTINUE
      IT=1
C
C
      WRITE BINARY (10) ICONTROL(7)   ;bel to reset.
C
      CALL BYTEPAC(ICONTROL(1))              ;send ESC-SUB
      CALL BYTEPAC(ICONTROL(3))              ;in one word
                                             ;so back in Gin.C
      GO TO 1000                             ;final position.
C
 1100 IT=0                   ;on receipt of both positions.
```

```
        DO 1110 IN=1,2
           IWORD2(IN)=IOR(IH(IN),IL(IN))                  ;form position
                                    ;in tekpoints for now.
 1110   CONTINUE
C
        ISTOP=IWORD1(1)-IWORD2(1)              ;check for stop
        IF(ISTOP.EQ.0)GO TO 2001               ;
C
C******     Calculate crosshair position (X only) in terms of
C       words, where 256 words = 1 block = 4 vectors = 32 msec.
C
        ISWORD=(ISTART*256) + (IWORD1(1)-30)*5 +
     /  IFIX(FLOAT(IWORD1(1)-30)/3)            ;
C
        INWORD=(IWORD2(1)-IWORD1(1))*5 +
     /  IFIX(FLOAT(IWORD2(1)-IWORD1(1))/3)     ;
C
C******     Continue filling file ISTORE with documentation.
C
        WRITE(3,1299)                          ;
 1299   FORMAT(11X,"-------------------------")
        WRITE(3,1300) ICOUNT                   ;
 1300   FORMAT(10X,"SPEECH SEGMENT NUMBER: ",I6)
        WRITE(3,1305) ISWORD                   ;
 1305   FORMAT(10X,"FIRST WORD HEARD WAS:  ",I6)
        WRITE(3,1310) INWORD                   ;
 1310   FORMAT(10X,"NUMBER OF WORDS HEARD: ",I6)
        RLENGTH= FLOAT(INWORD / 8000.)         ;length sec.
        WRITE(3,1315) RLENGTH
 1315   FORMAT(10X,"SEGMENT LENGTH (SEC): ",F7.4)
C
C******     Calculate number of times segment will
C       be heard via the program TALK.SV.
C
        IHEARD= 10*(IAND(IMASK(3),IHEAR1))     ;ten's position.
        IHEARD= IHEARD + (IAND(IMASK(3),IHEAR2))  ;entire #.
C
C******     This is where control is transferred over to
C       program TALK.SV, in order to hear the speech.  The
C       transfer is accomplished using a "CALL SWAP".
C
        PARAM(8) = ISWORD                      ;These are the para-
        PARAM(9) = INWORD                      ;meters used by
        PARAM(10) = IHEARD                     ;TALK.SV for outputting
                                               ;speech.
        CALL CFILW("PARAM.AD",2,IER5)
        IF(IER5 .NE. 1) TYPE"CFILW ERROR = ",IER5
        CALL OPEN(2,"PARAM.AD",3,IER6)
        IF(IER6 .NE. 1) TYPE"OPEN ERROR = ",IER6
        CALL WRSEQ(2,PARAM,20,IER7)
        IF(IER7 .NE. 1) TYPE"WRSEQ ERROR = ",IER7
        CALL CLOSE(2,IER8)
        IF(IER8 .NE. 1) TYPE"CLOSE ERROR = ",IER8
        CALL SWAP("TALK.SV",IER9)
```

```
      IF(IER9 .NE. 1) TYPE"SWAP ERROR = ",IER9
C
C******     On return from the speech output subroutine,
C       continue in the interactive loop and increment counter.
C
      ICOUNT = ICOUNT + 1
      GO TO 990
 2001 CONTINUE
C
C******     On receipt of interrupt command form Tektronix;
C       notify user that program is waiting for input, then
C       continue as directed.
C
      DO 2010 IWAIT=1,4                          ;get the
         ACCEPT" <7> <7> <7> "                          ;attention of
         CALL FDELAY(20)                                ;the user.
 2010 CONTINUE
      TYPE" <15> <15> "
      CALL BYTEPAC(ICONTROL(1))
      CALL BYTEPAC(ICONTROL(2))
      CALL FDELAY(20)
C
C******     In case user accidently sent the same position twice
C       this portion of the program will allow continuation of
C       work.
C
      TYPE"      INTERRUPT COMMAND RECEIVED FROM TEKTRONIX <15> "
      TYPE"      Enter 0 : to STOP."
      ACCEPT"      Enter 1 : to CONTINUE IN SAME MODE <15>",IFLAG
      IF(IFLAG.EQ.0)GO TO 2020
      ISAMP=0
      IBYTE=999
      CALL BYTEPAC(IBYTE)
      CALL RESET
      CALL OPEN(5,FILENAM,1,512,IER1)
      OPEN 19,"DCOMPS",LEN=ISIZE
      IF(IER1 .NE. 1) TYPE"EROR ON OPEN, IER1= ",IER1
      OPEN 1,"TENERGY"
      OPEN 4,"IENERGY"
      GO TO 31
 2020 CALL RESET
      STOP
      END
```

```
C*****************************************************************************
C     Title: Talk
C     Author: Lt Allen
C     Date: Dec 82

C     Compile command:
C        FORTRAN TALK

C     Load commands:
C        RLDR/P 2/K TALK SAMCONFIG4 DSAMLIB@

C     Function:
C        This program plays back specified sections of a speech file
C        on the Eclipse A/D/A device. It reads from file PARAM.AD
C        the filename, beginning word, number of words, and number
C        of times to repeat playback. This program is intended to
C        be used by having the user program first write the parameters
C        to file and then swapping in this program. The code used to
C        drive the Eclipse A D/A device is marked " <-- " and should
C        not be altered without knowledge of how the device operates

C*****************************************************************************

      EXTERNAL ID321     ; <--
      EXTERNAL ID323     ; <--
      COMMON / IBUFF / IWASTE(1024)    ; <--
      COMMON / IBUF0 / IDATA3(15360)   ; <--

      INTEGER IDRBA(16)    ; <--
      INTEGER PARAM(10),FIRBLK,NUMBLK,SKIP

      CALL FOPEN(1,"PARAM.AD")
      CALL RDSEQ(1,PARAM,20,IER)
      IF (IER.NE.1) TYPE "RDSEQ error ",IER
      CALL FCLOSE(1)

      FIRBLK=INT((PARAM(8)-1)/256)
      IF (PARAM(9).LE.256) NUMBLK=1
      IF (PARAM(9).GT.256) NUMBLK=INT(PARAM(9)/256)+1
      SKIP=PARAM(8)-((FIRBLK*256)+1)

      CALL OPEN(1,PARAM,1,IER)
      IF (IER.NE.1) TYPE "OPEN error ",IER," with your file"
      CALL RDBLK(1,FIRBLK,IDATA3,NUMBLK,IER)
      IF (IER.NE.1) TYPE "RDBLK error ",IER," with your file"
      CALL CLOSE(1,IER)
      IF (IER.NE.1) TYPE "CLOSE error ",IER," with your file"


      IDATA1=64000K     ; <--
      IDATA2=PARAM(9)   ; <--
```

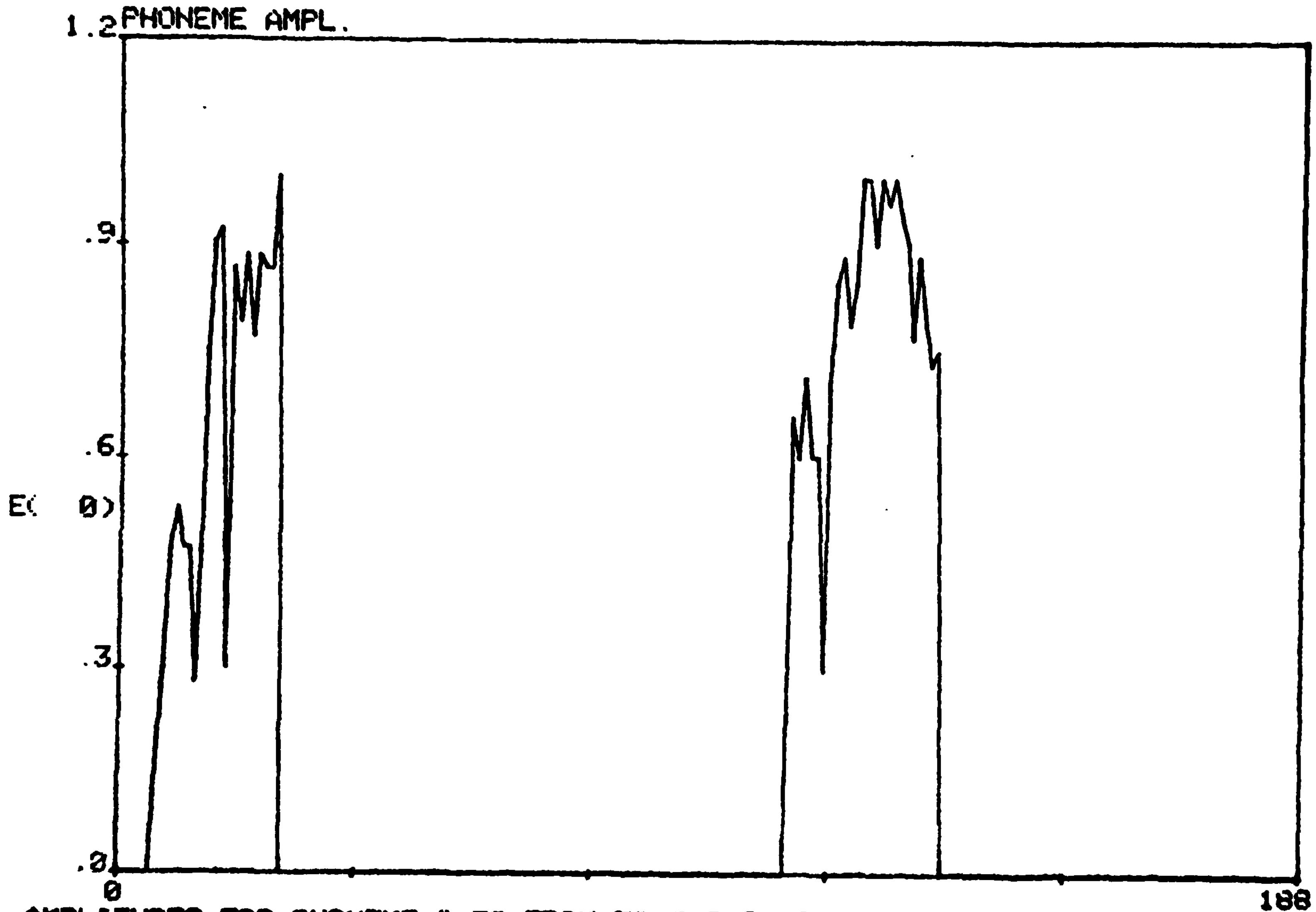
```
      CALL DSTRT(IER)    ; <--
      IF (IER.NE.1) CALL ERROR("DSTRT error")    ; <--

      I=1
      J=SKIP+1

10    IDATA3(I)=IDATA3(J)
      I=I+1
      J=J+1
      IF (I.LE.PARAM(9)) GO TO 10

      K=0
15    CALL DOITW(IIRSA,ID323,8,IDATA1,IDATA2,IDATA3,IER)    ; <--
      IF (IER.NE.1) TYPE "DOITW error ",IER                 ; <--

      K=K+1
      CALL WAIT(1,2,IER)
      IF (IER.NE.1) TYPE "WAIT error ",IER
      IF (K.LT.PARAM(10)) GO TO 15

      CALL EXIT
      END
```

APPENDIX B

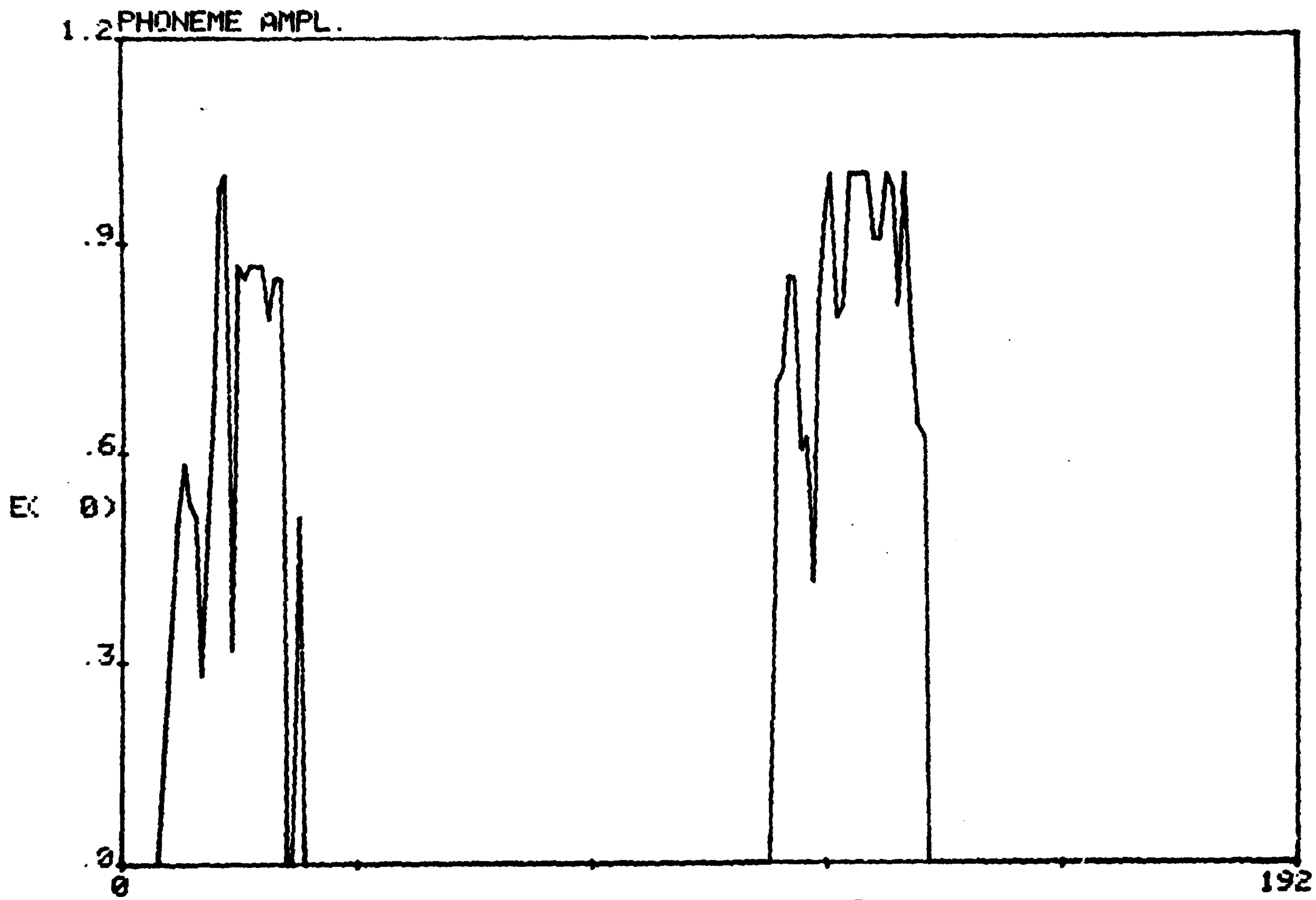
PHONEME AMPLITUDE
PLOTS

B. Phoneme Amplitude Plots

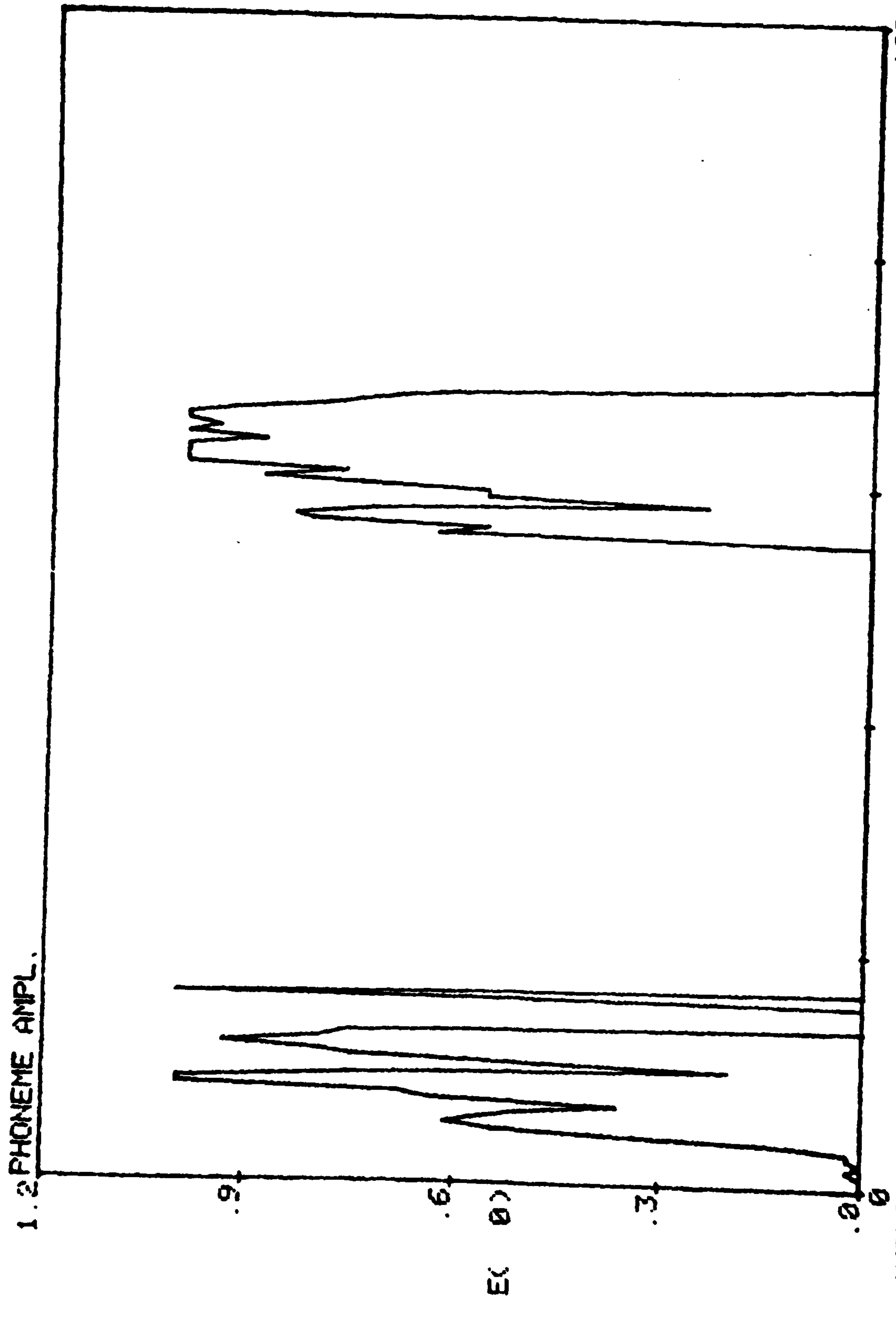
The following plots are a representative cross section of the total population of phonemes that showed a degradation in performance. A sample is only used because with 29 phonemes showing a degraded performance, this could account for approximately 145 plots.



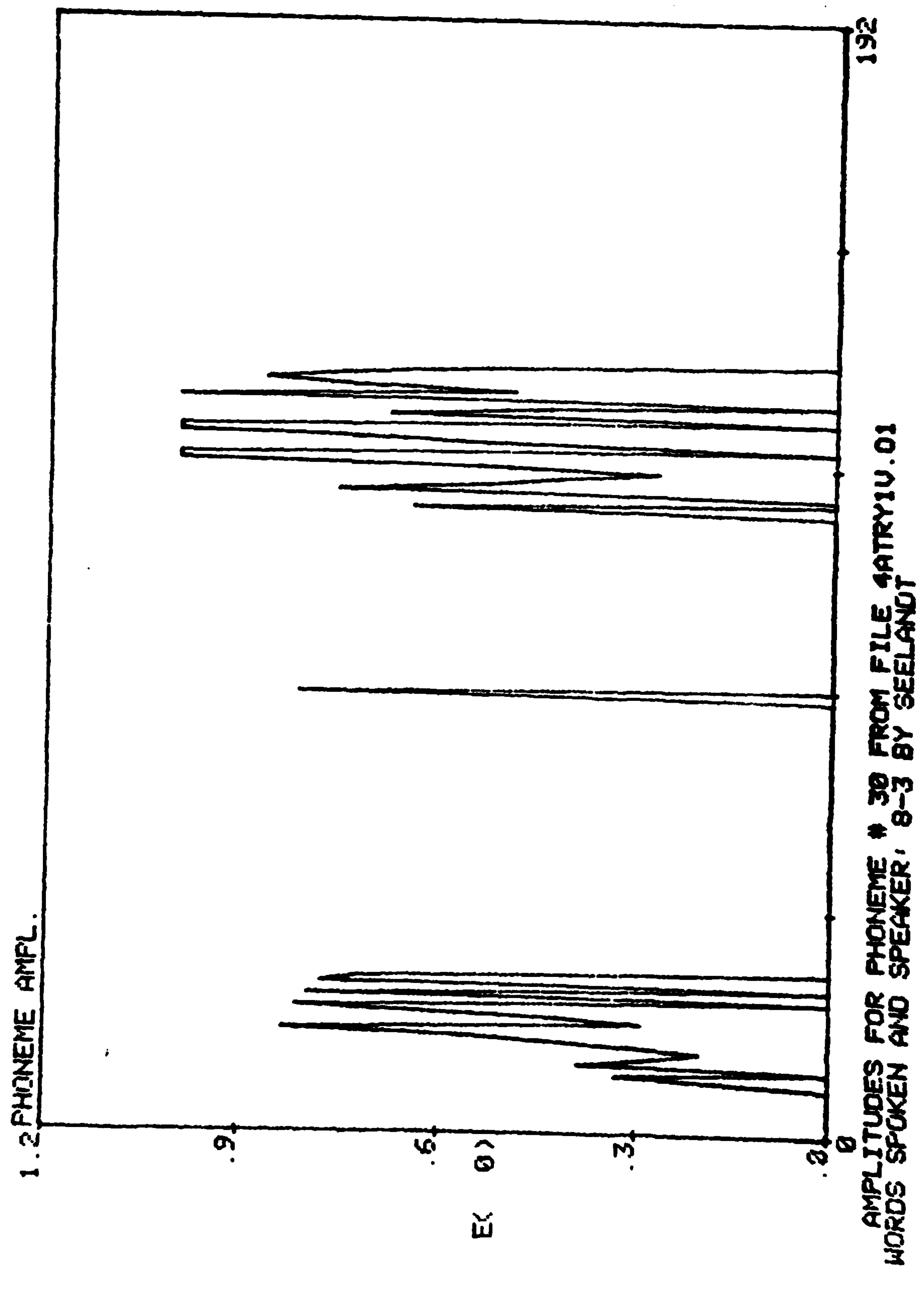

AMPLITUDES FOR PHONEME # 30 FROM FILE F8P4ASV.01
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT

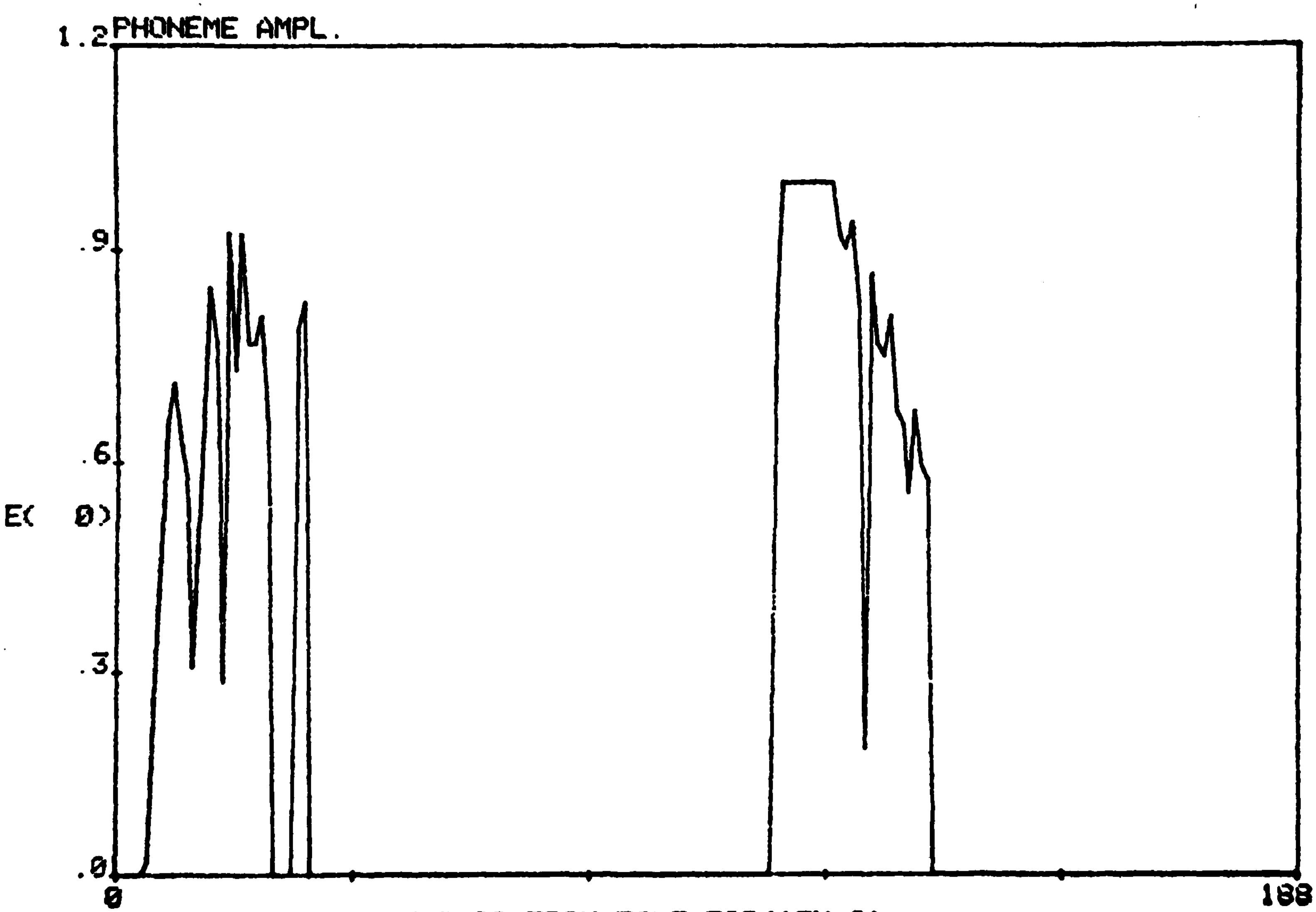



AMPLITUDES FOR PHONEME # 30 FROM FILE 3VFSP4A.01
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT

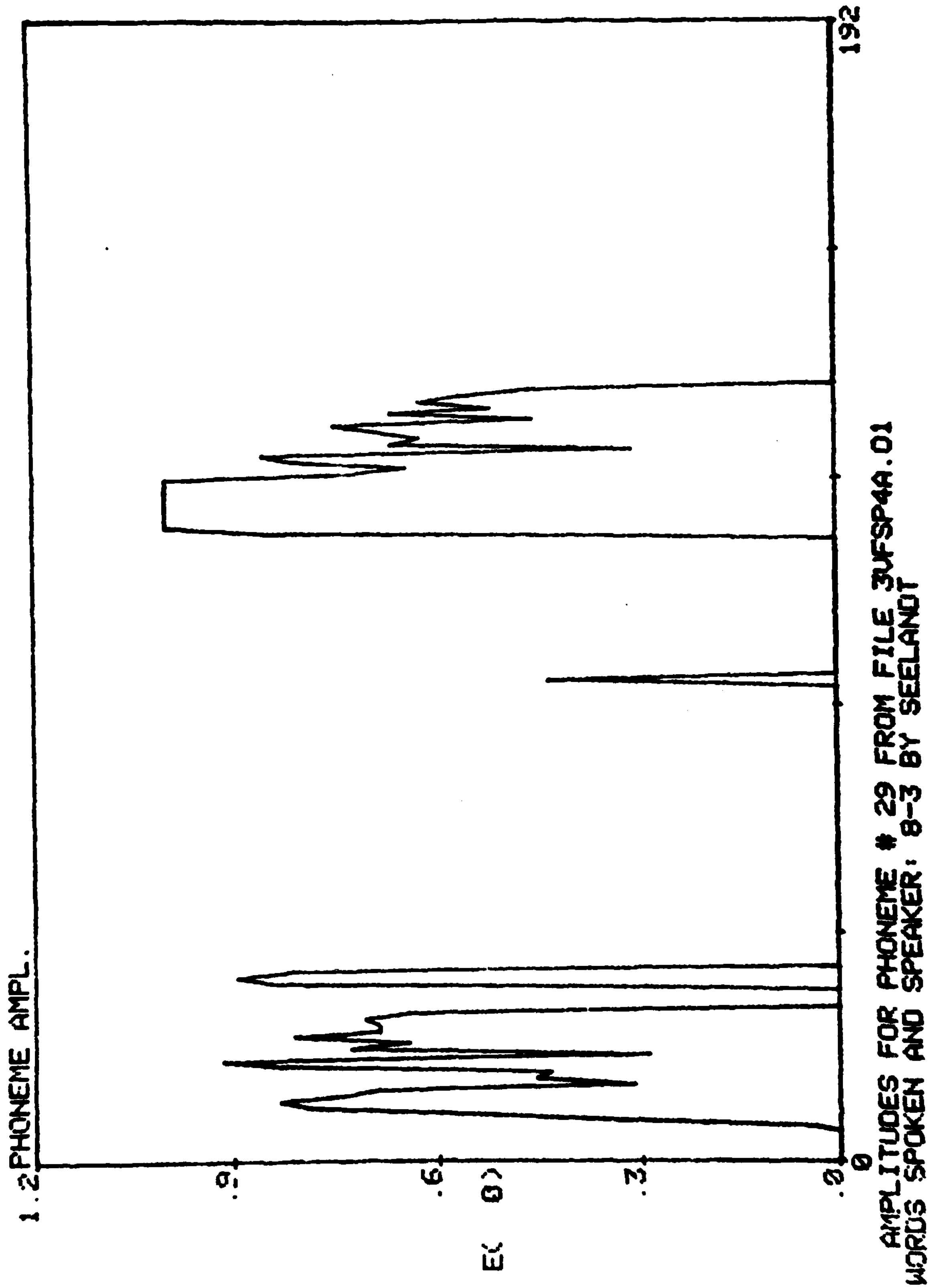



AMPLITUDES FOR PHONEME # 30 FROM FILE 4ATRY2F.01
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT

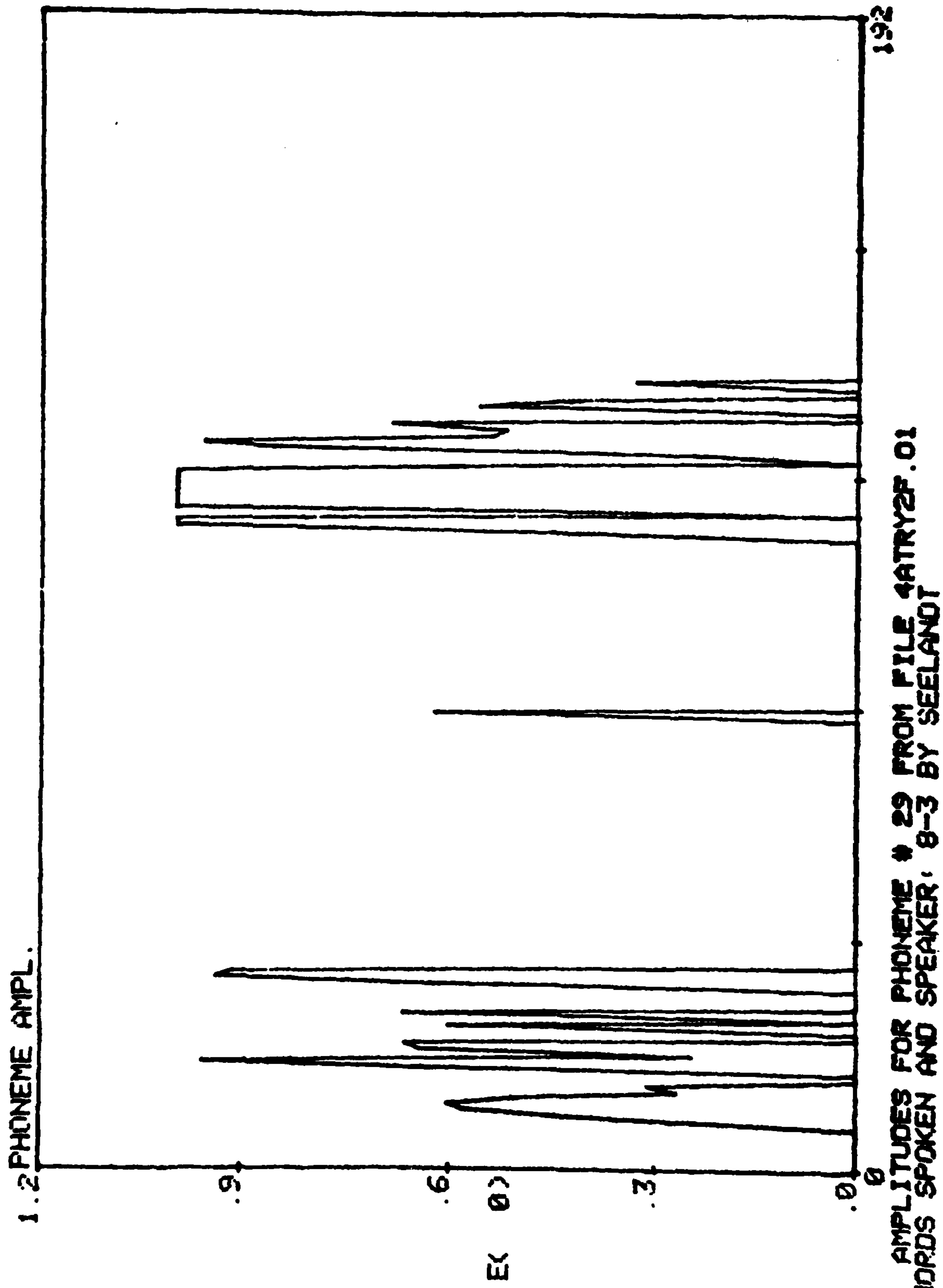

PHONEME AMPL.
1.2
.9
.6
.3
.0
0
192
AMPLITUDES FOR PHONEME # 30 FROM FILE 4ATRY1V.01
WORDS SPOKEN AND SPEAKER; 8-3 BY GEELANDT



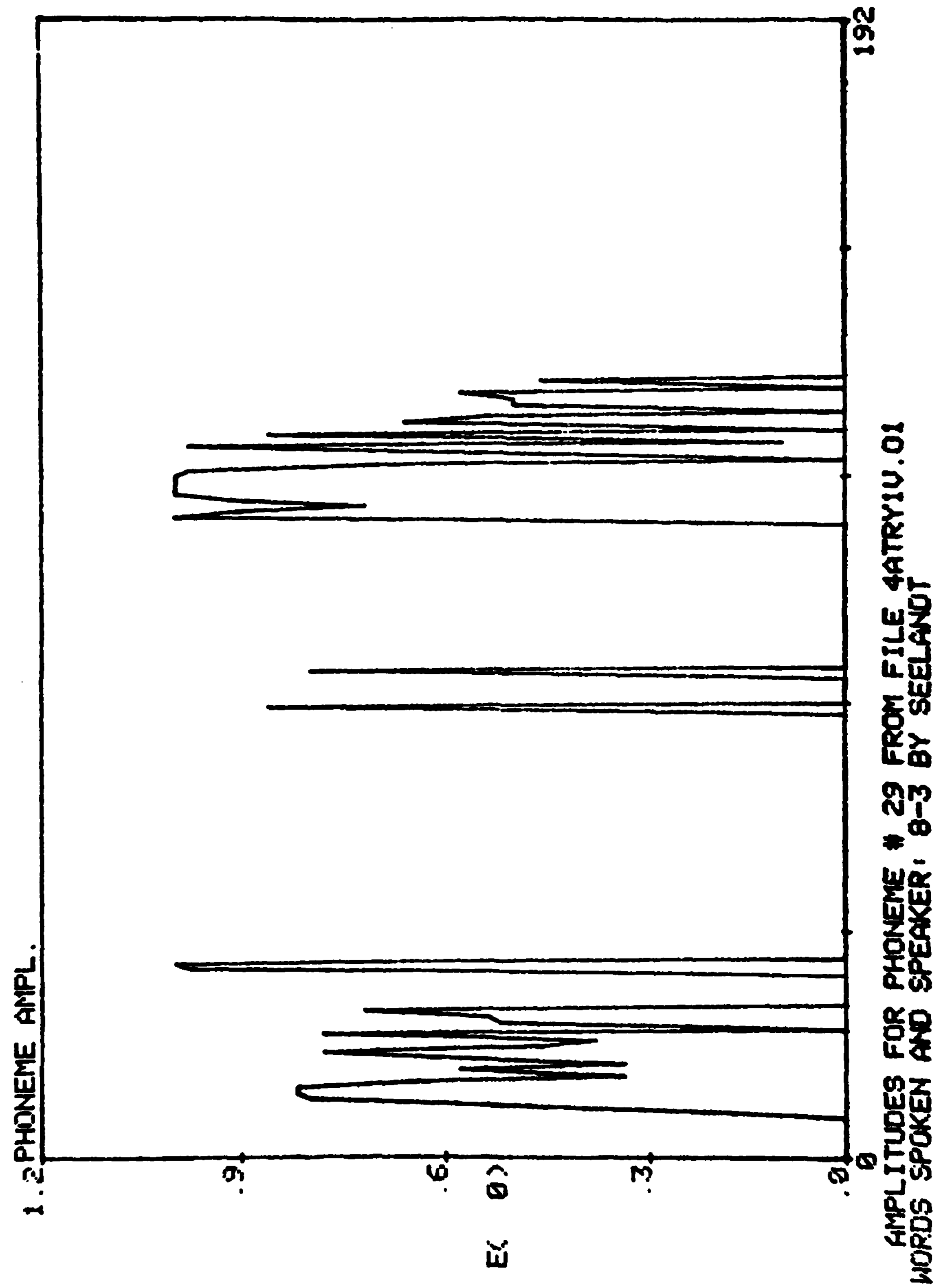

AMPLITUDES FOR PHONEME # 29 FROM FILE FSP4A5V.01
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT

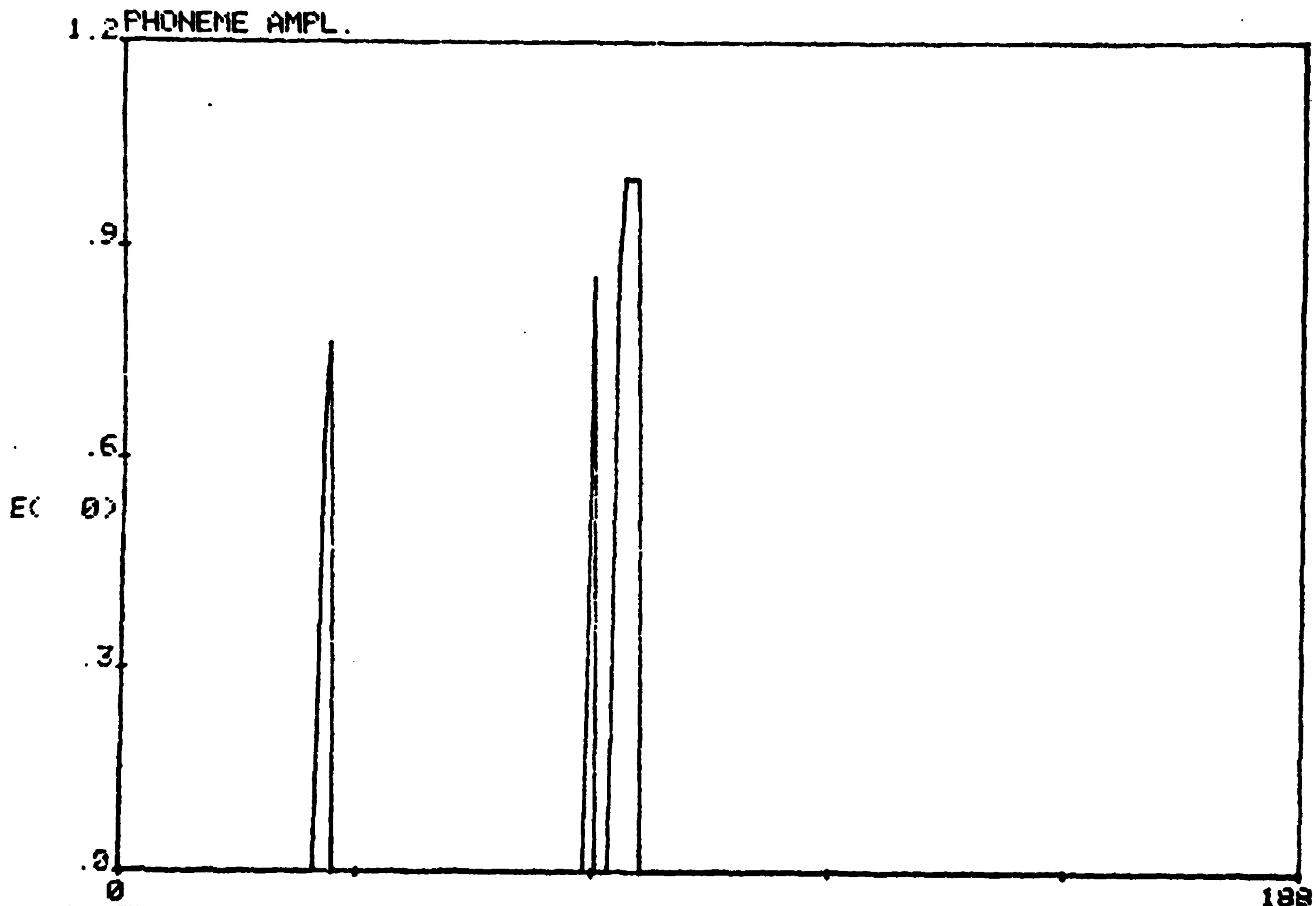


PHONEME AMPL.
1.2
.9
.6
.3
.0
E( 0)
0
192
AMPLITUDES FOR PHONEME # 29 FROM FILE 3WFSP4A.D1
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT

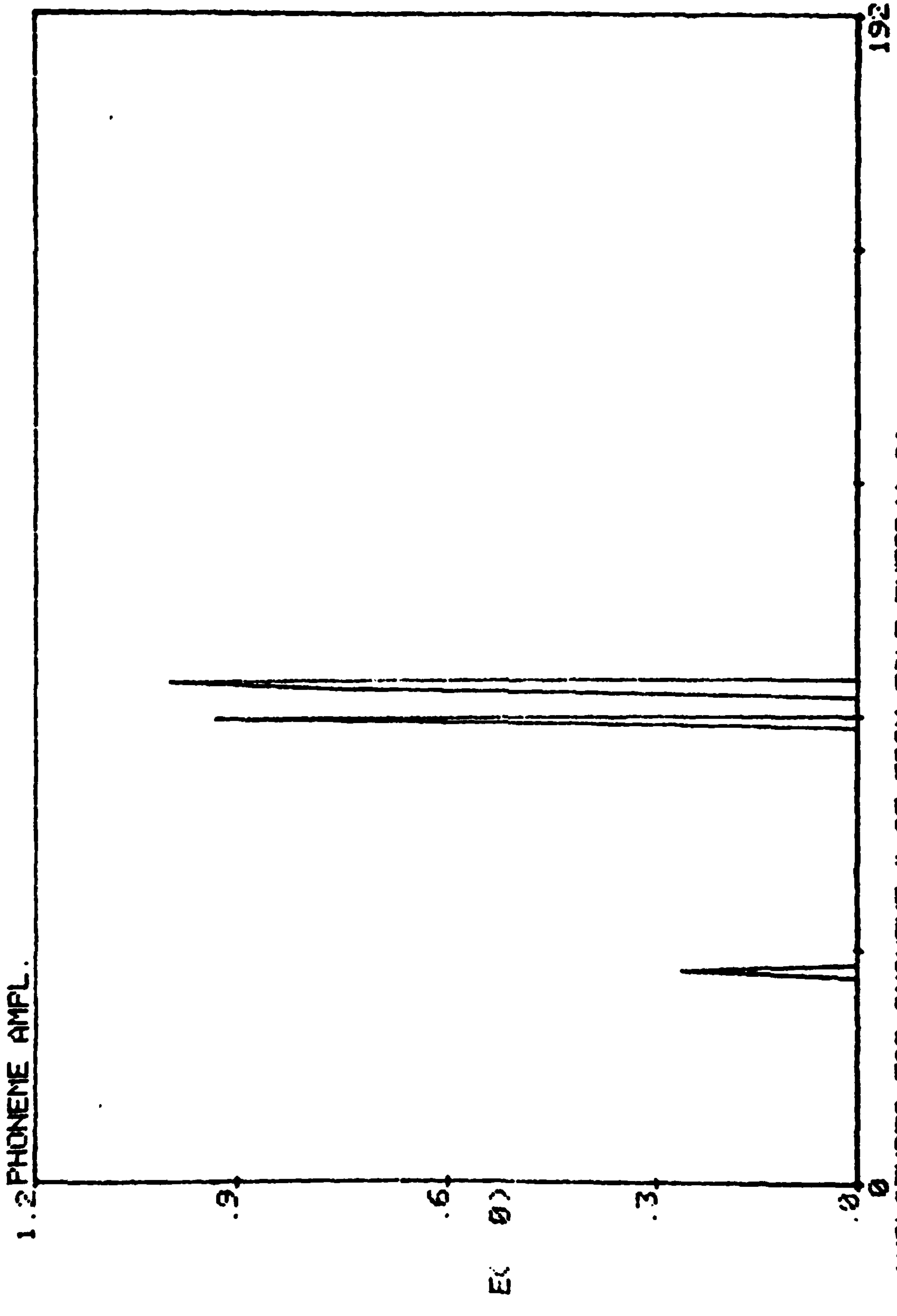


PHONEME AMPL.
1.2
.9
.6
.3
.0
E( 0)
0
192
AMPLITUDES FOR PHONEME # 29 FROM FILE 4ATRY2F.01
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT

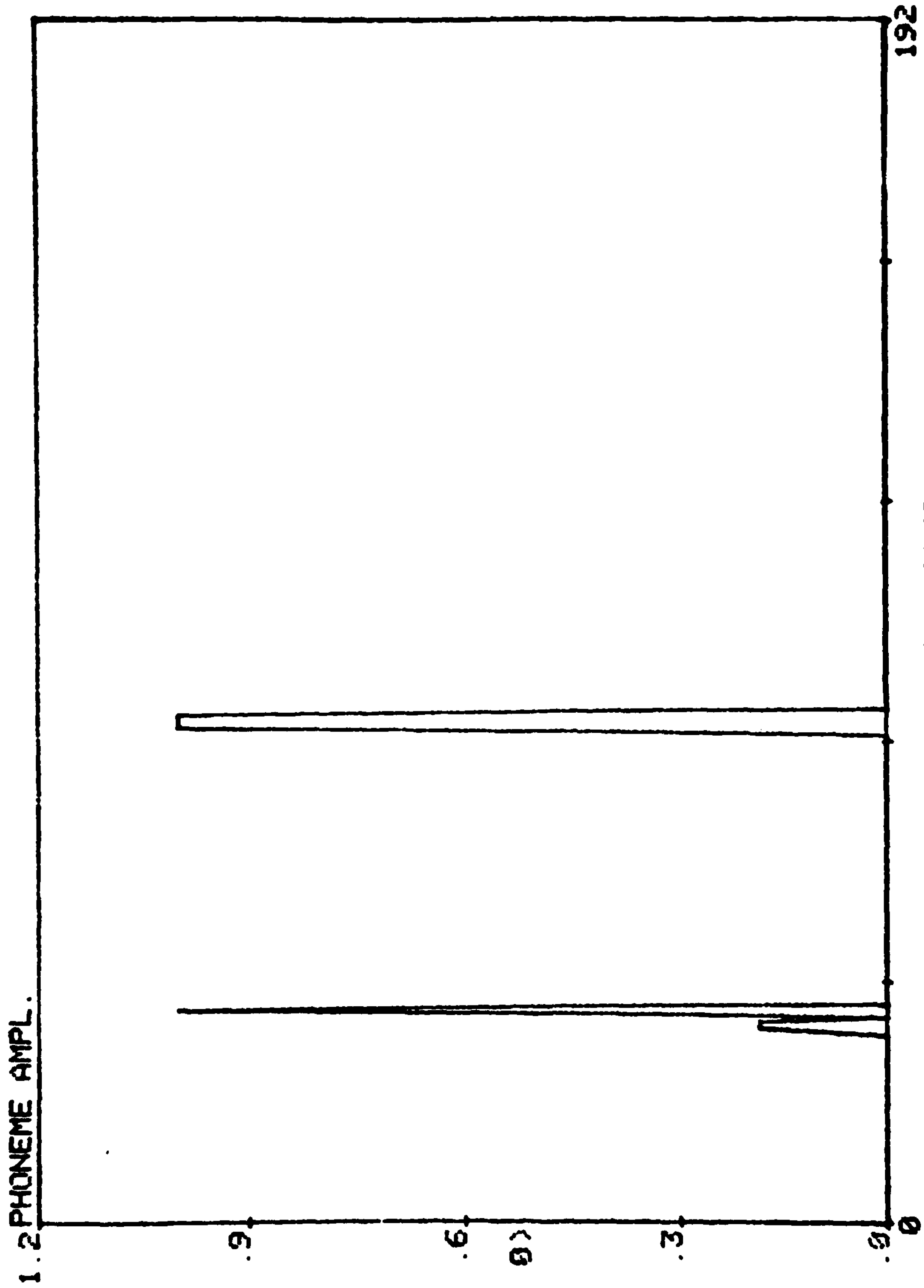

PHONEME AMPL.
1.2
.9
.6
.3
.0
E( 0)
0
192
AMPLITUDES FOR PHONEME # 29 FROM FILE 4ATRY1V.01
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT



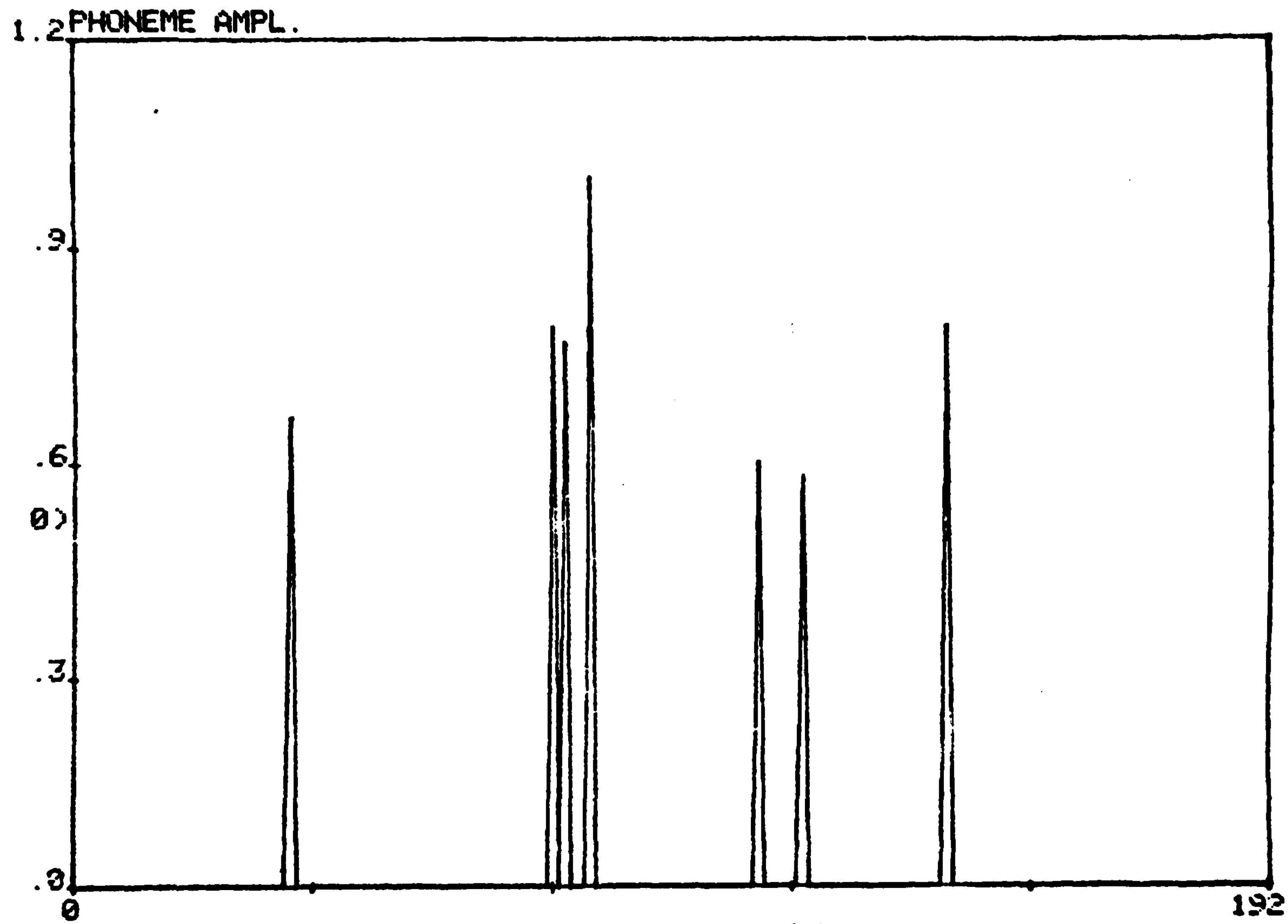

AMPLITUDES FOR PHONEME # 25 FROM FILE FSP4A5V.01
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT



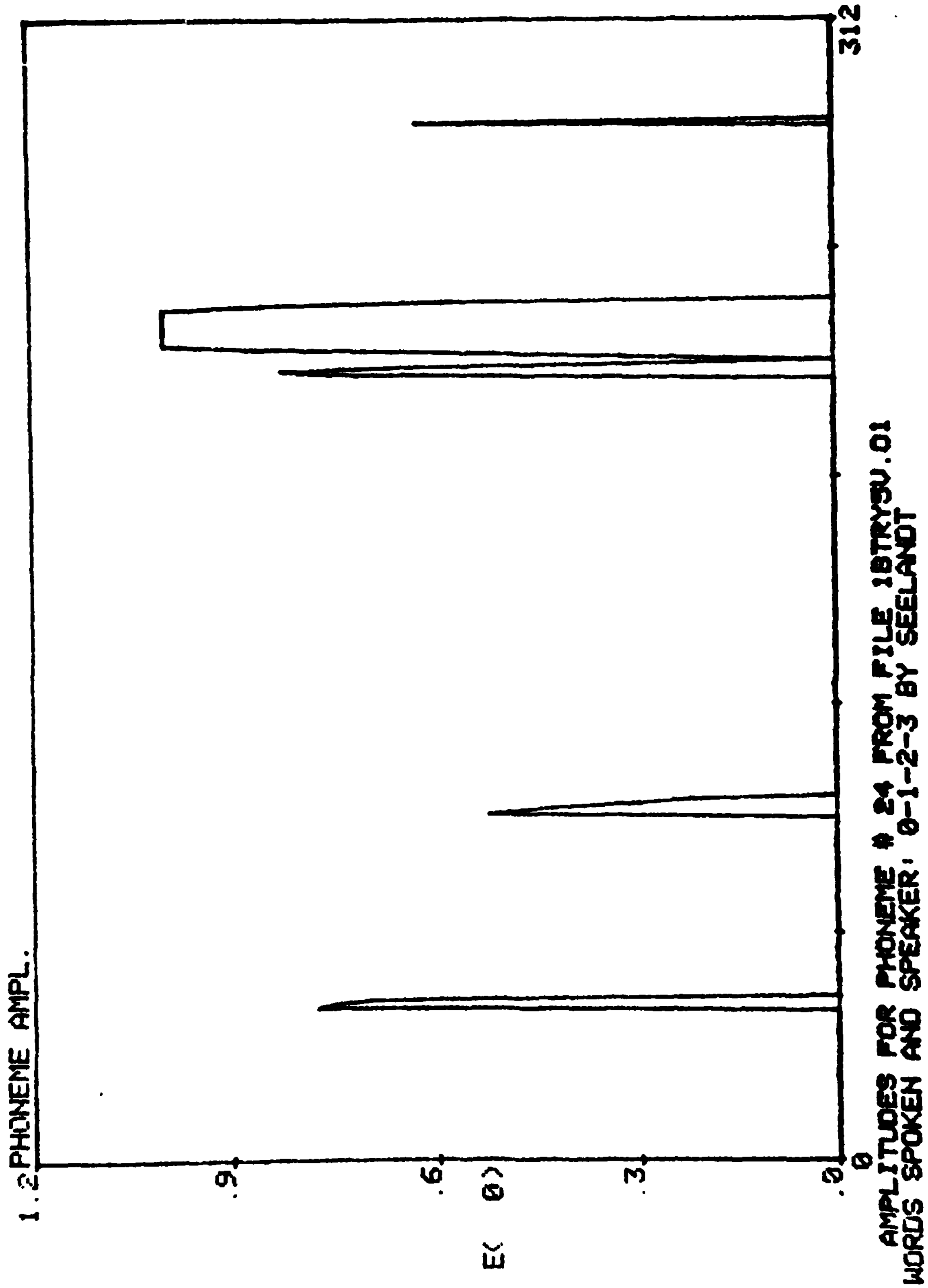

AMPLITUDES FOR PHONEME # 23 FROM FILE 3VFSP4A.01
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT

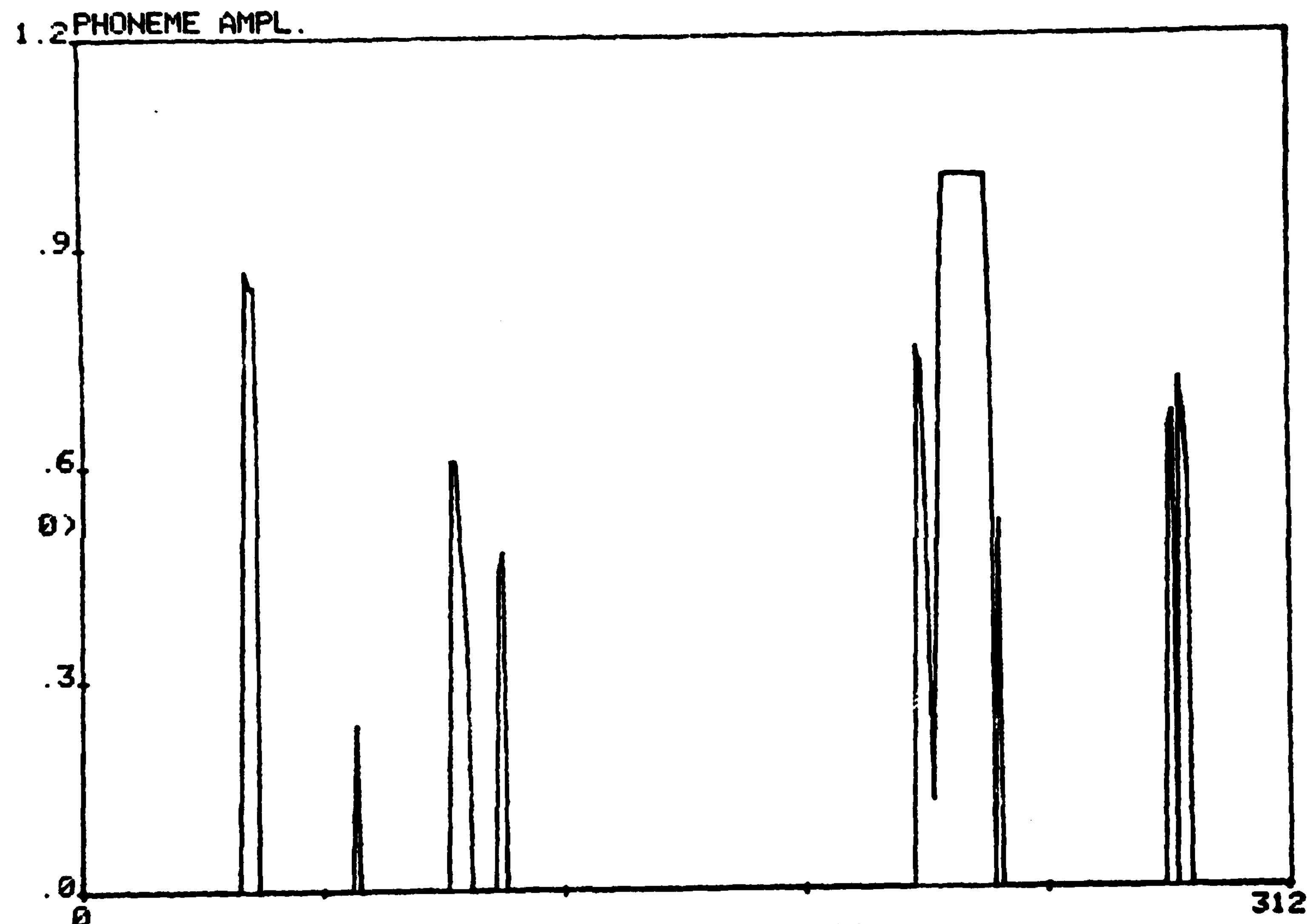



AMPLITUDES FOR PHONEME # 25 FROM FILE 4ATRY2F.01
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT

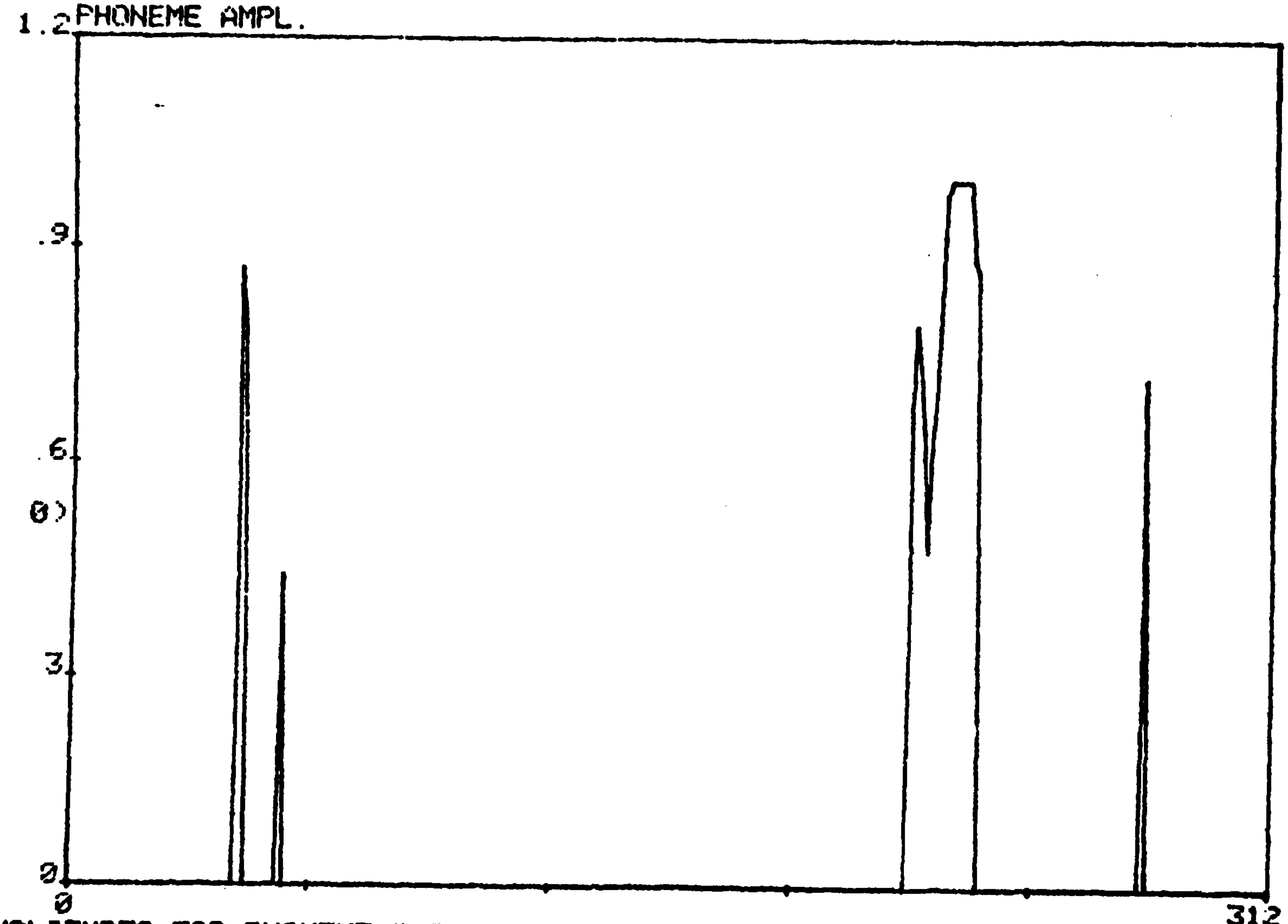


AMPLITUDES FOR PHONEME # 25 FROM FILE 4ATRY1V.01
WORDS SPOKEN AND SPEAKER: 8-3 BY SEELANDT

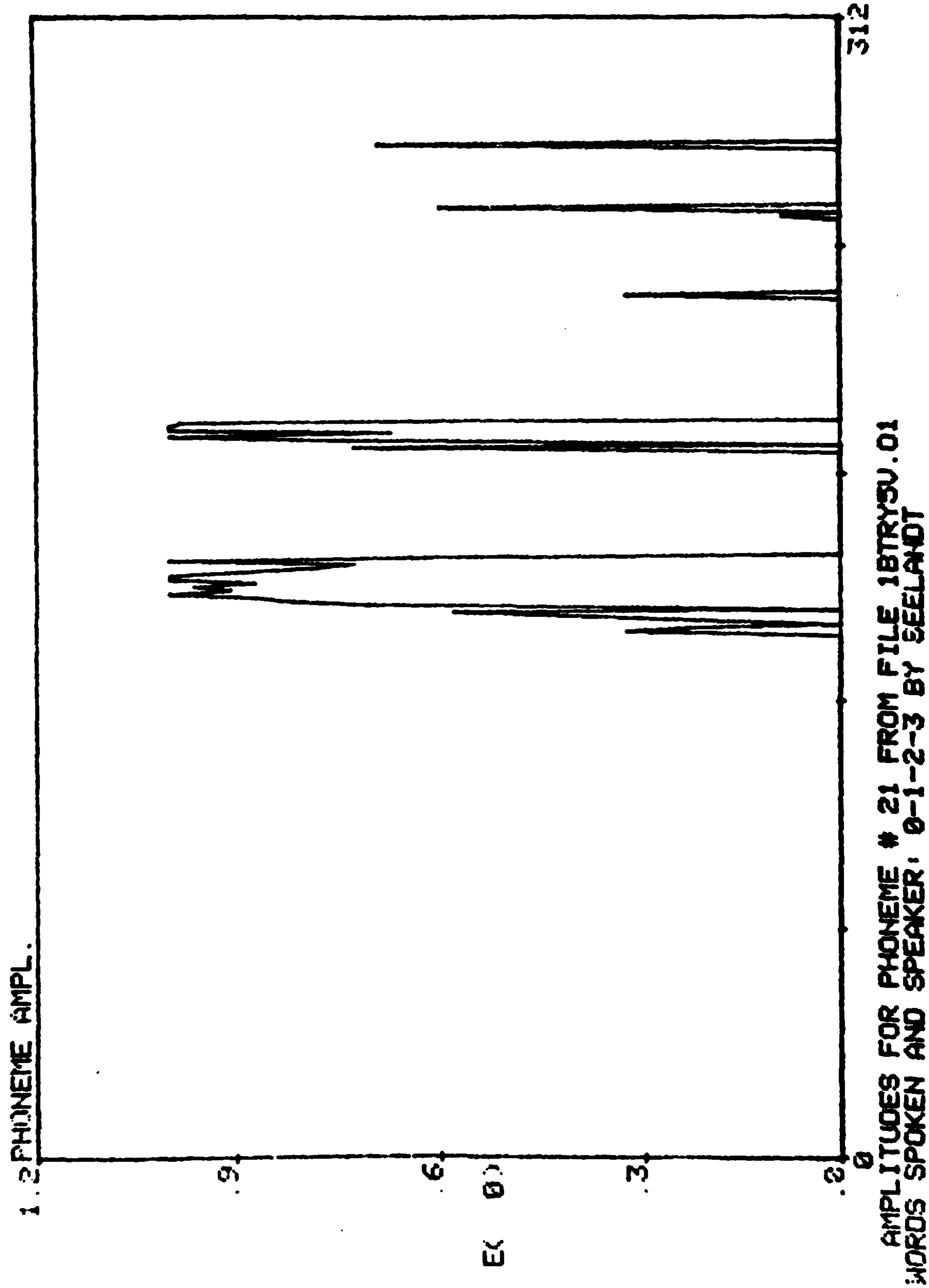

PHONEME AMPL.
1.2
.9
.6
.3
.0
E( 0)
0
312
AMPLITUDES FOR PHONEME # 24 FROM FILE 1BTRYSU.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



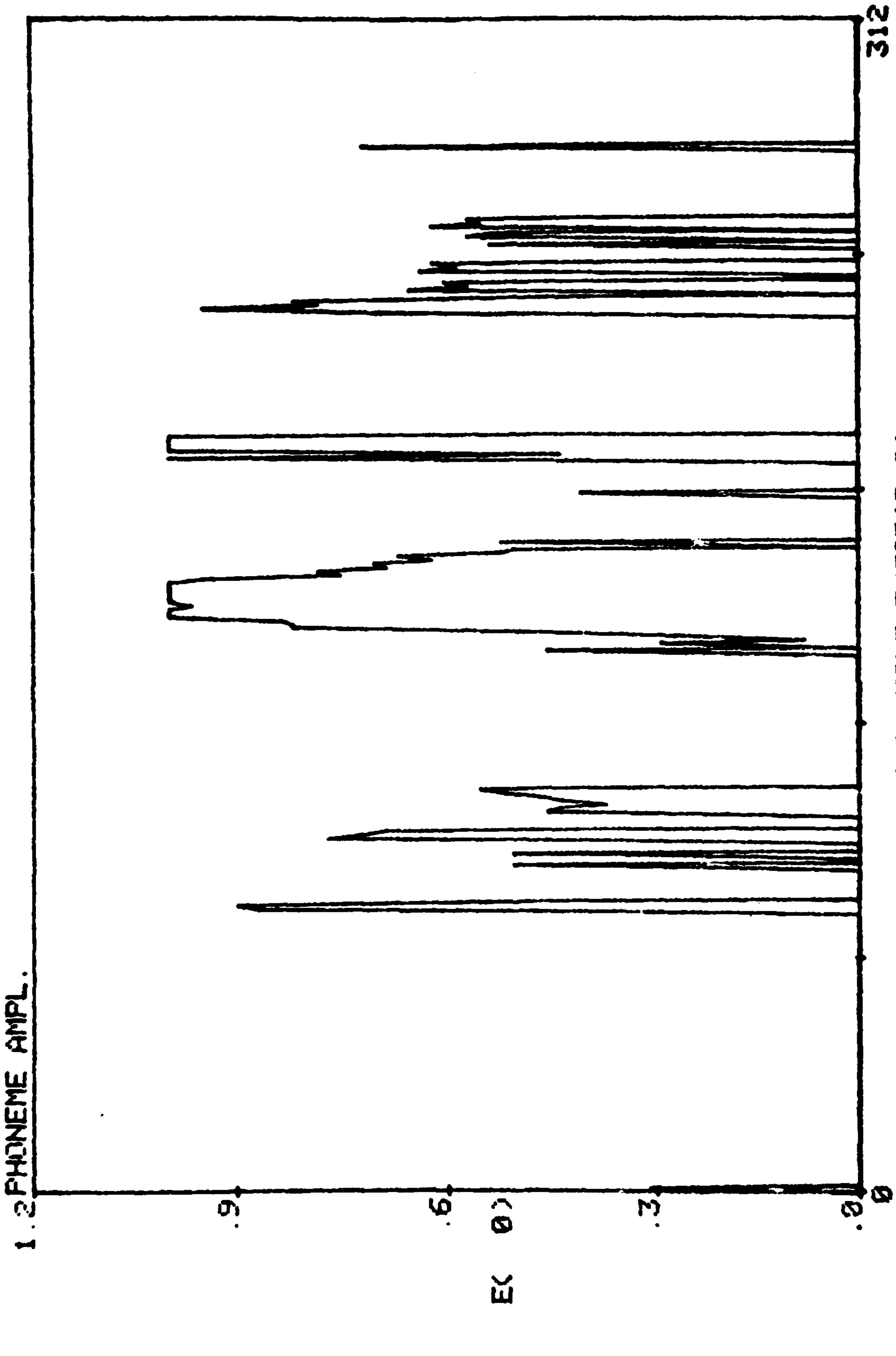

AMPLITUDES FOR PHONEME # 24 FROM FILE 3VFSP1B.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



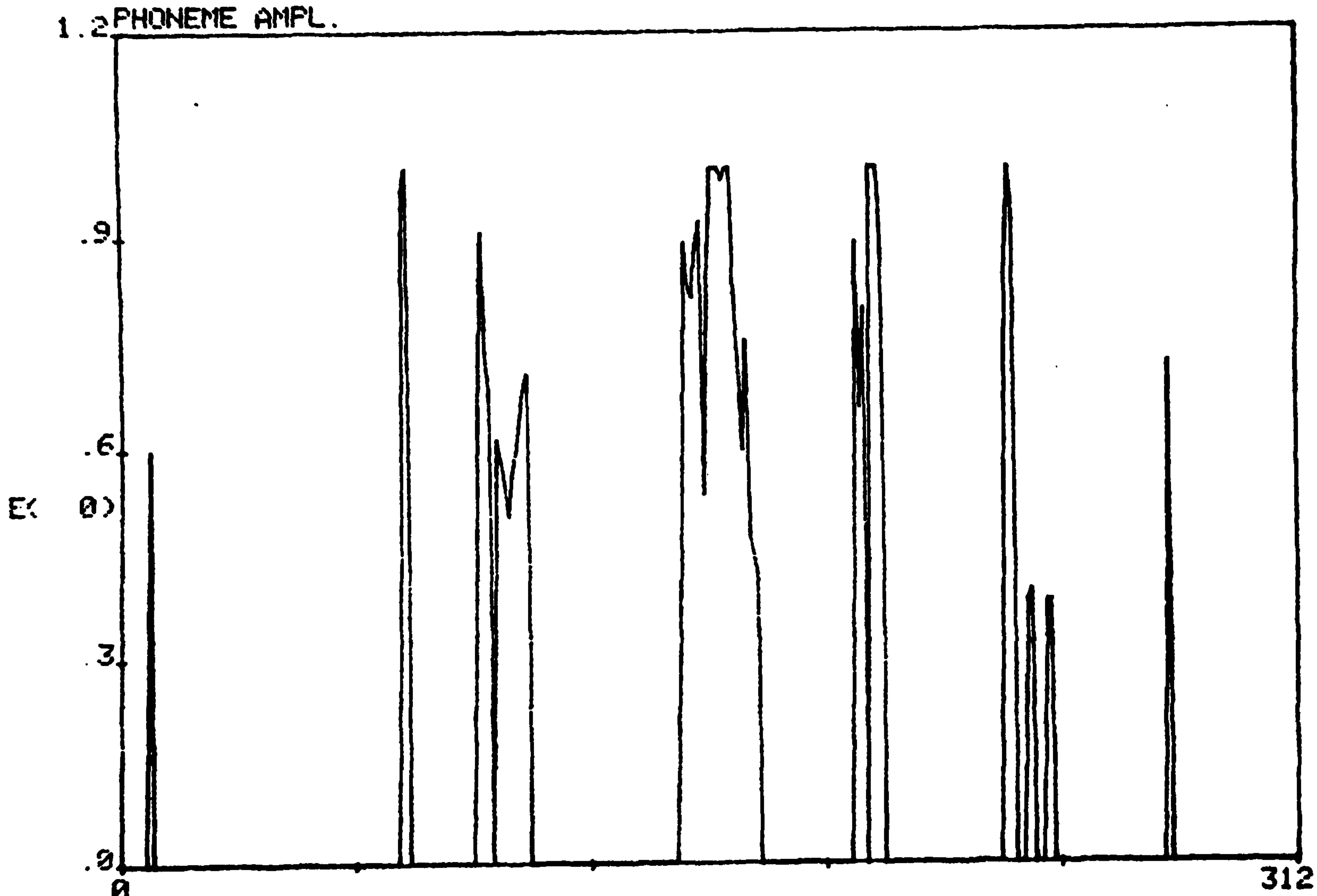

AMPLITUDES FOR PHONEME # 24 FROM FILE 1BTRY2F.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



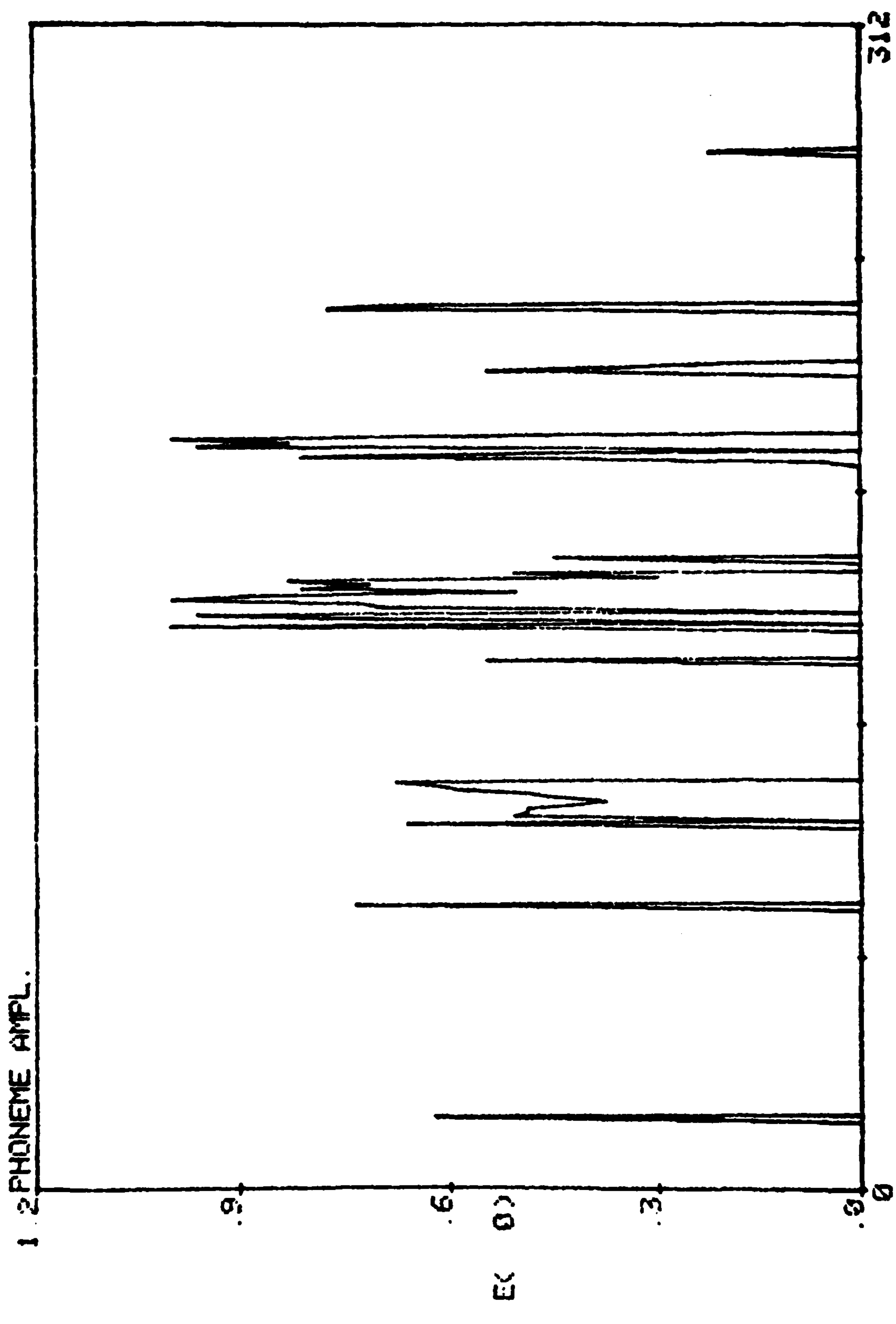
PHONEME AMPL.
1.2
.9
.6
.3
.0
0
312
E( 0)
AMPLITUDES FOR PHONEME # 21 FROM FILE 1BTRYSV.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



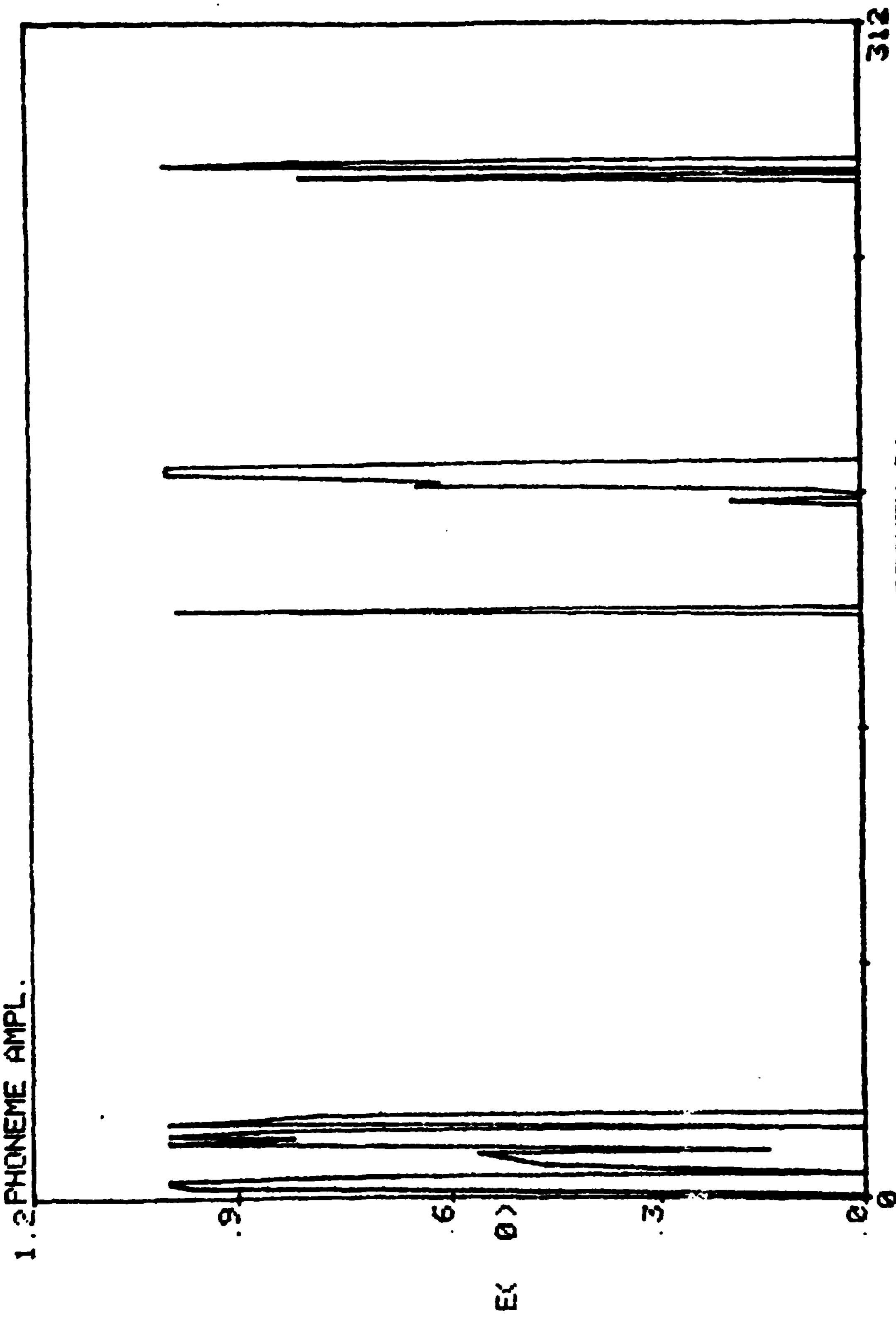

AMPLITUDES FOR PHONEME # 21 FROM FILE 3UFSP1B.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



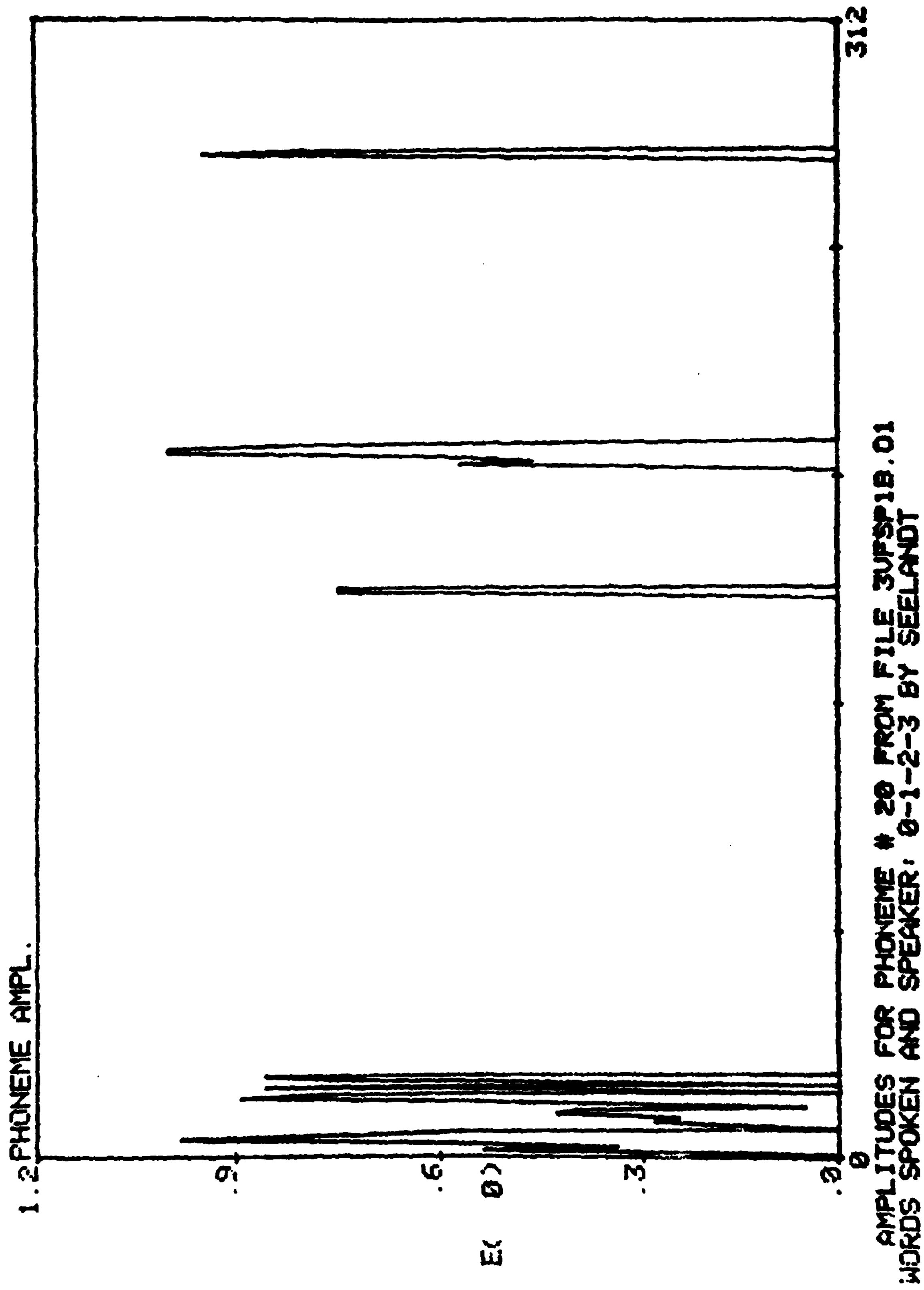

AMPLITUDES FOR PHONEME # 21 FROM FILE 1BTRY2F.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT

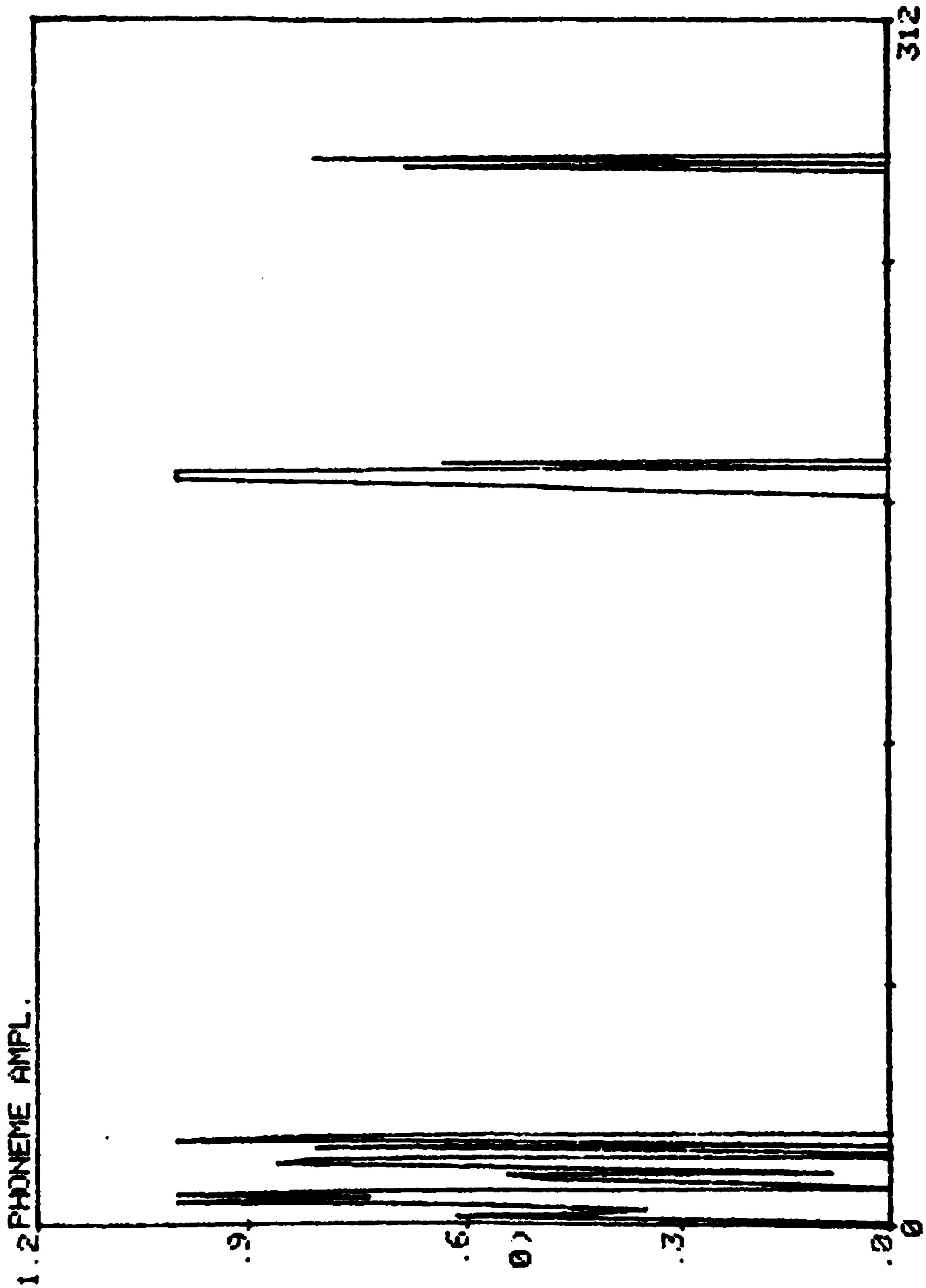



AMPLITUDES FOR PHONEME # 21 FROM FILE 1BTRY1V.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT

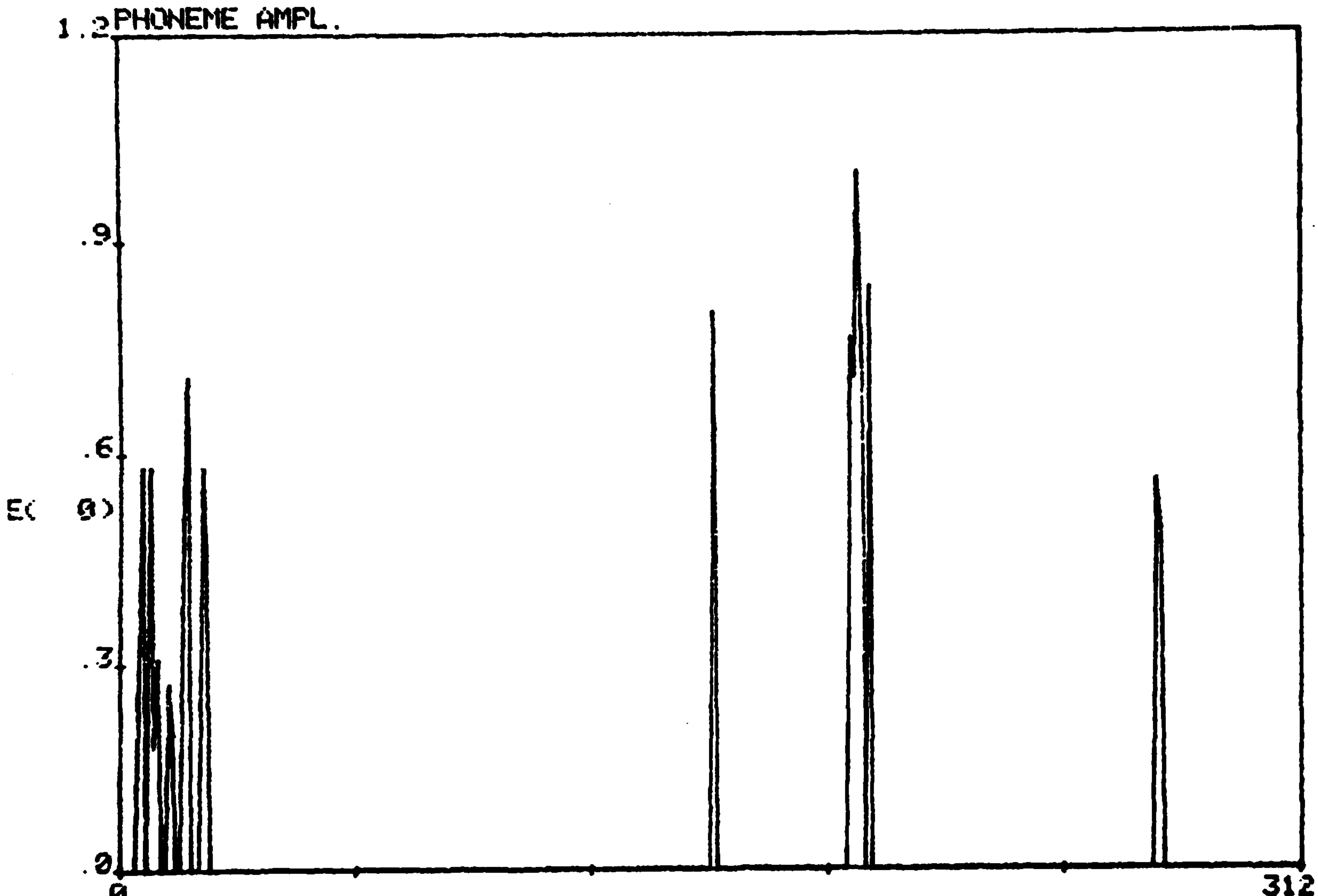



AMPLITUDES FOR PHONEME # 20 FROM FILE 1BTRYSV.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT


PHONEME AMPL.
1.2
.9
.6
.3
.0
E( 0)
0
312
AMPLITUDES FOR PHONEME # 20 FROM FILE 3UPSP1B.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



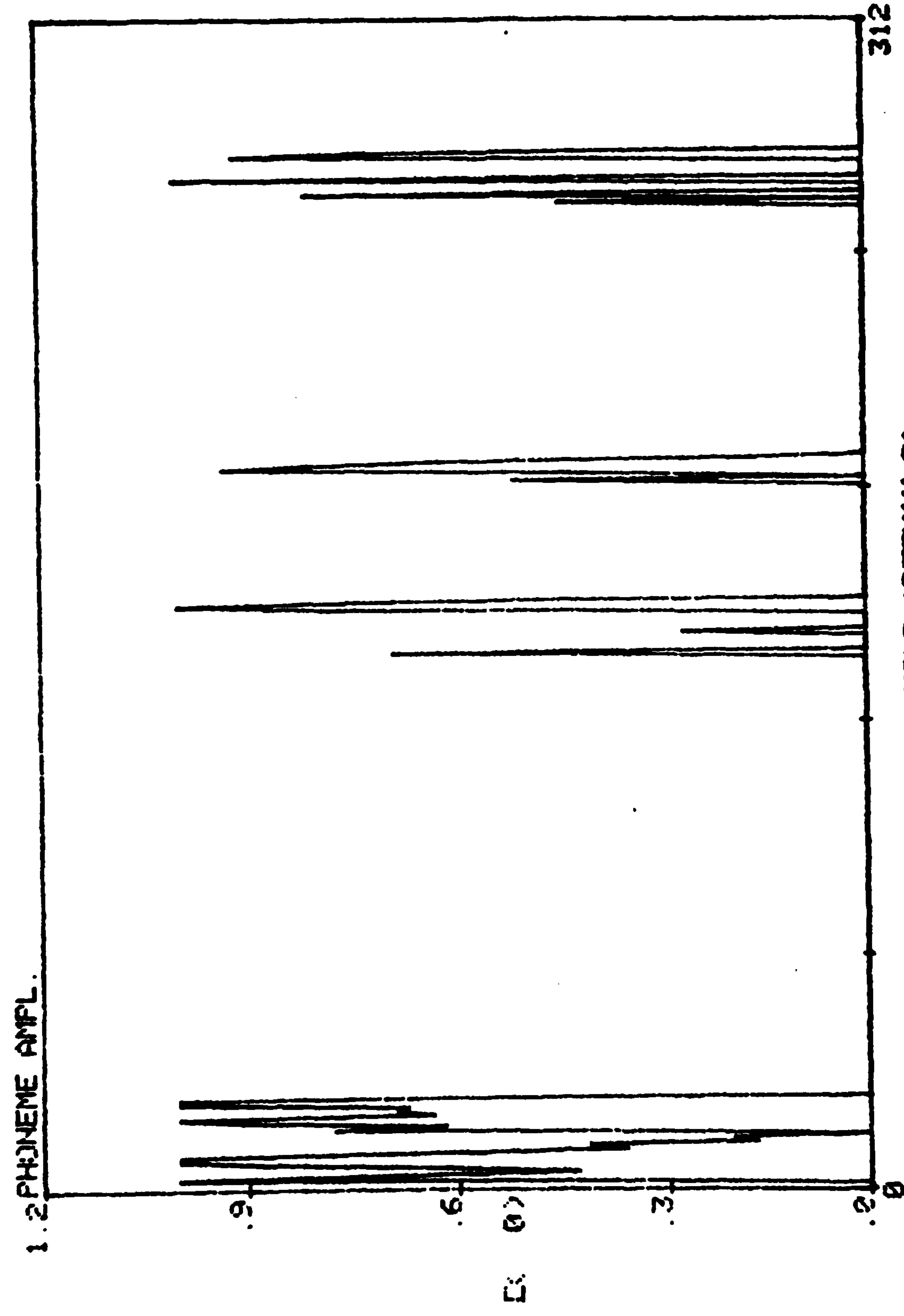
AMPLITUDES FOR PHONEME # 20 FROM FILE 1BTRY2F.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



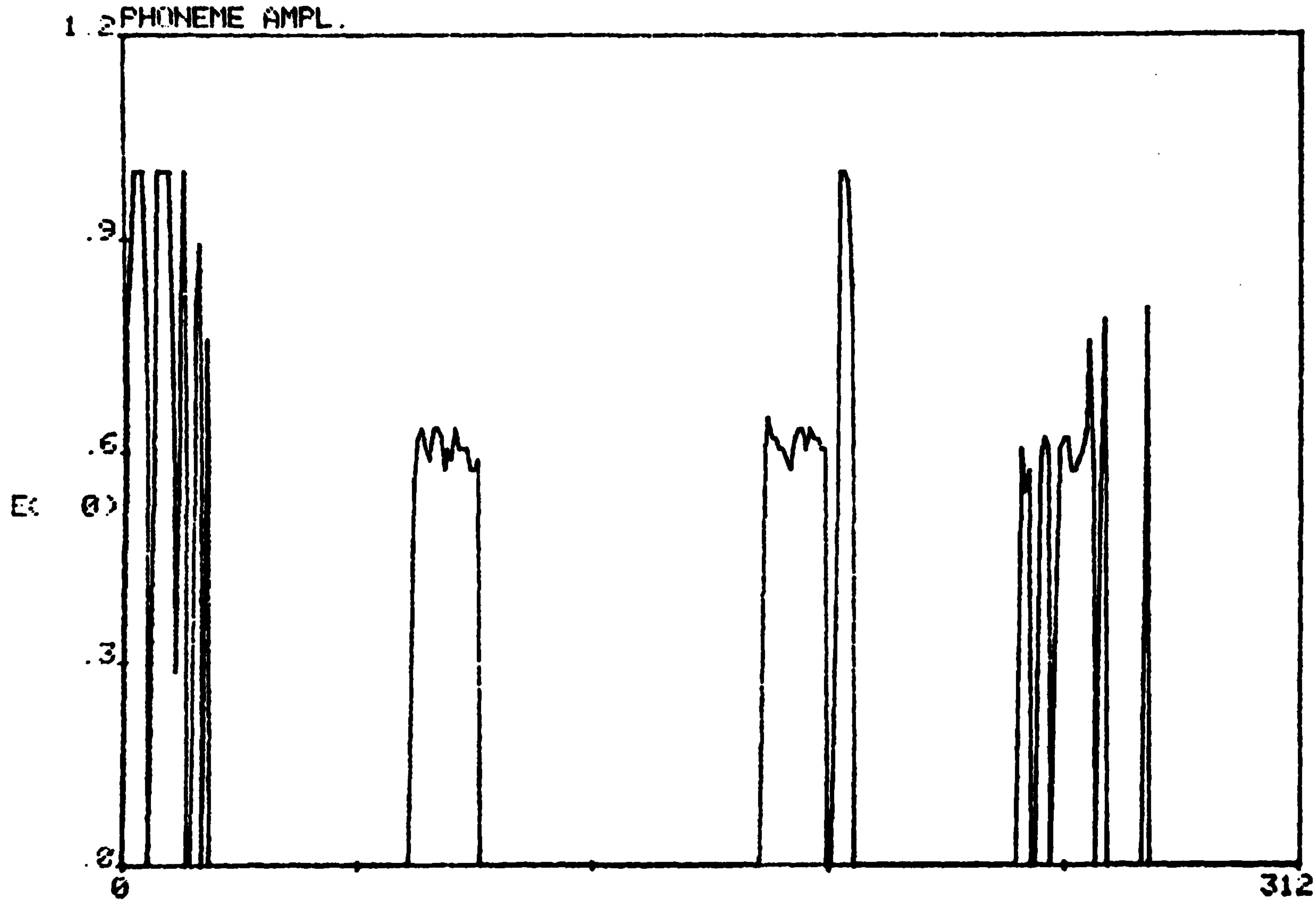
AMPLITUDES FOR PHONEME # 20 FROM FILE 1BTRY1V.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT

# APPENDIX C

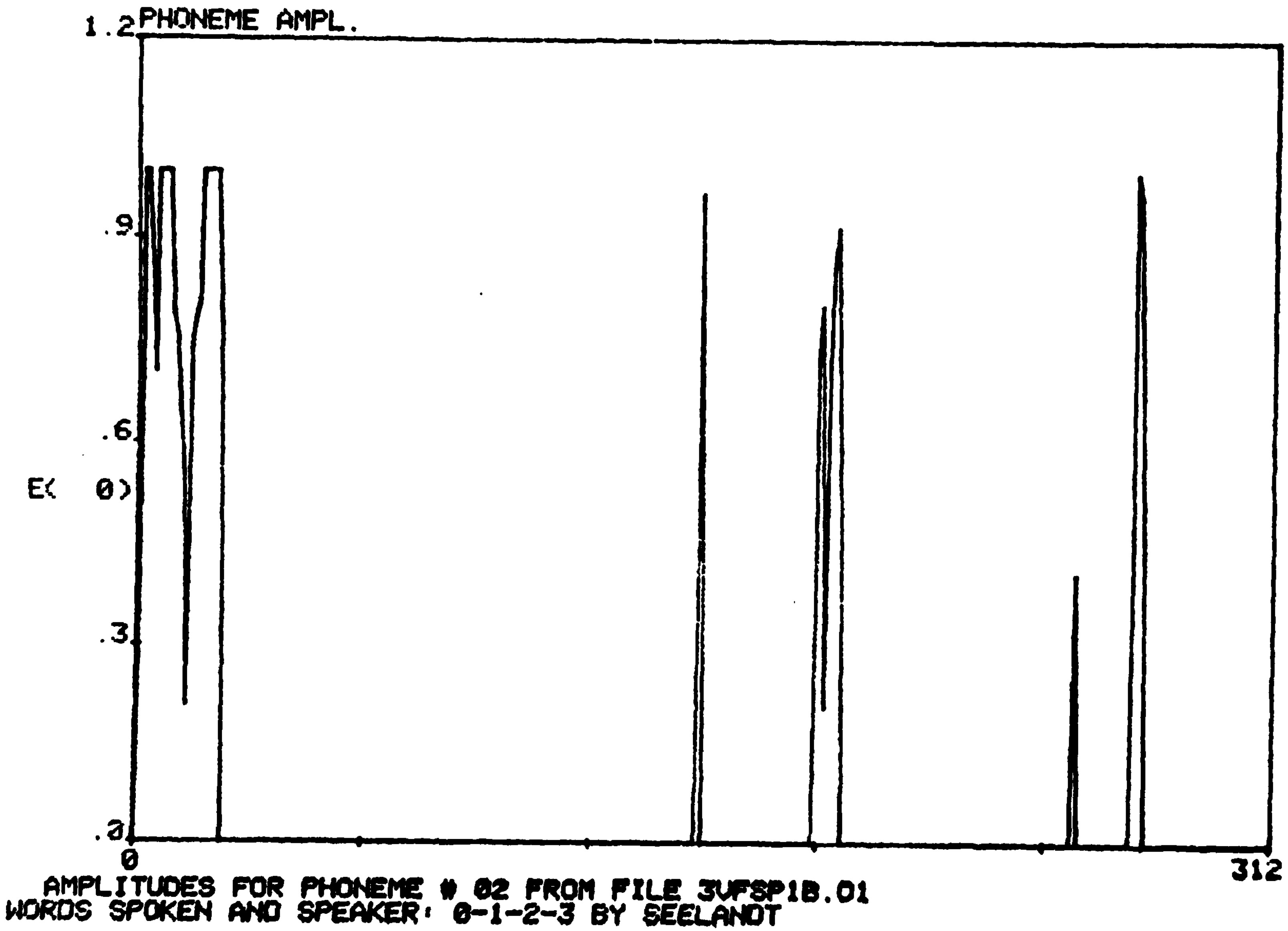
## PLOTTING THE SPECTRAL COMPONENTS OF A SPEECHFILE

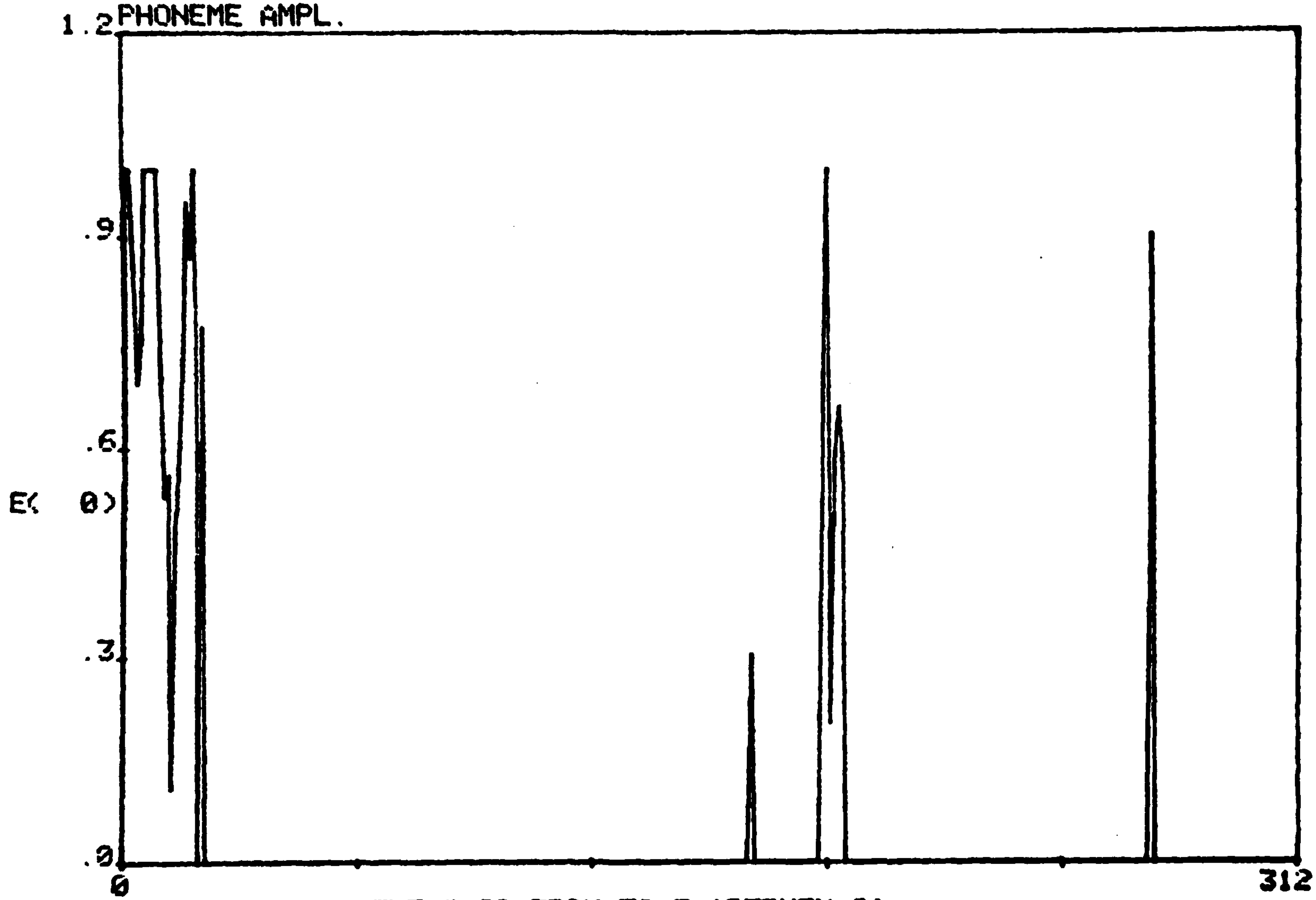



AMPLITUDES FOR PHONEME # 02 FROM FILE 1BTRY1V.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT

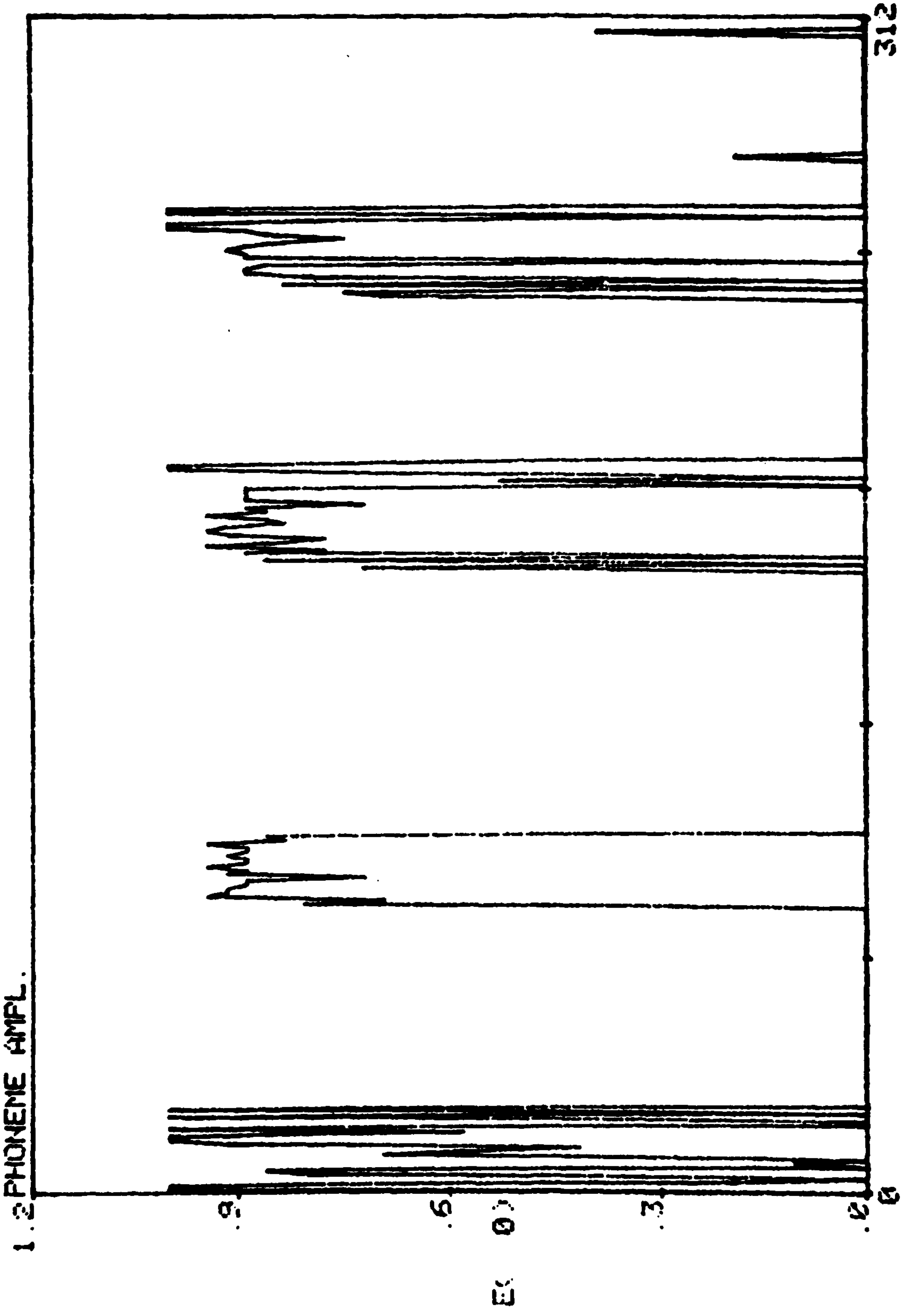



AMPLITUDES FOR PHONEME # 02 FROM FILE 1BTRY2F.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT

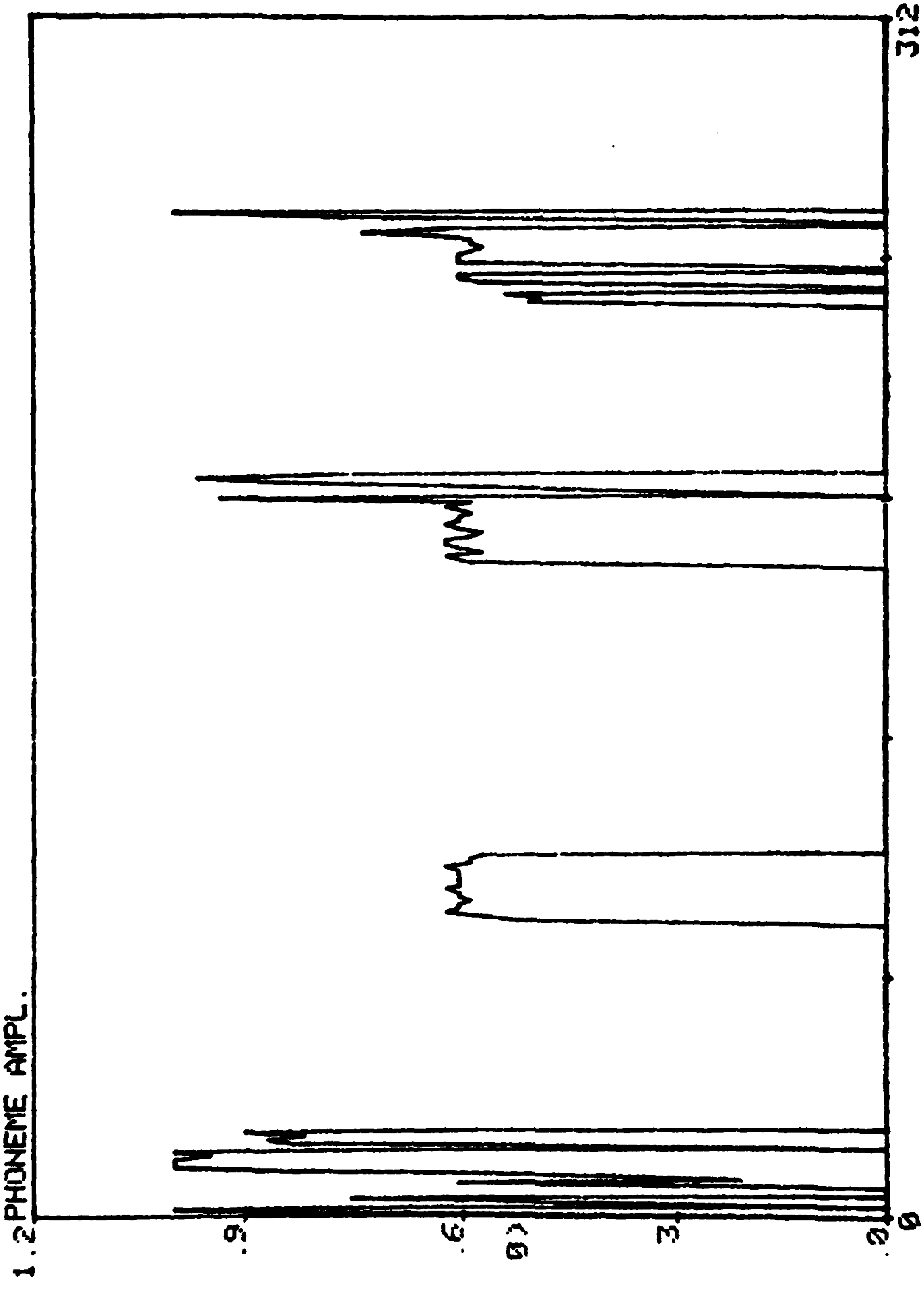

1.2 PHONEME AMPL.
.9
.6
E( 0)
.3
.0
0
312
AMPLITUDES FOR PHONEME # 02 FROM FILE 3VFSP1B.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



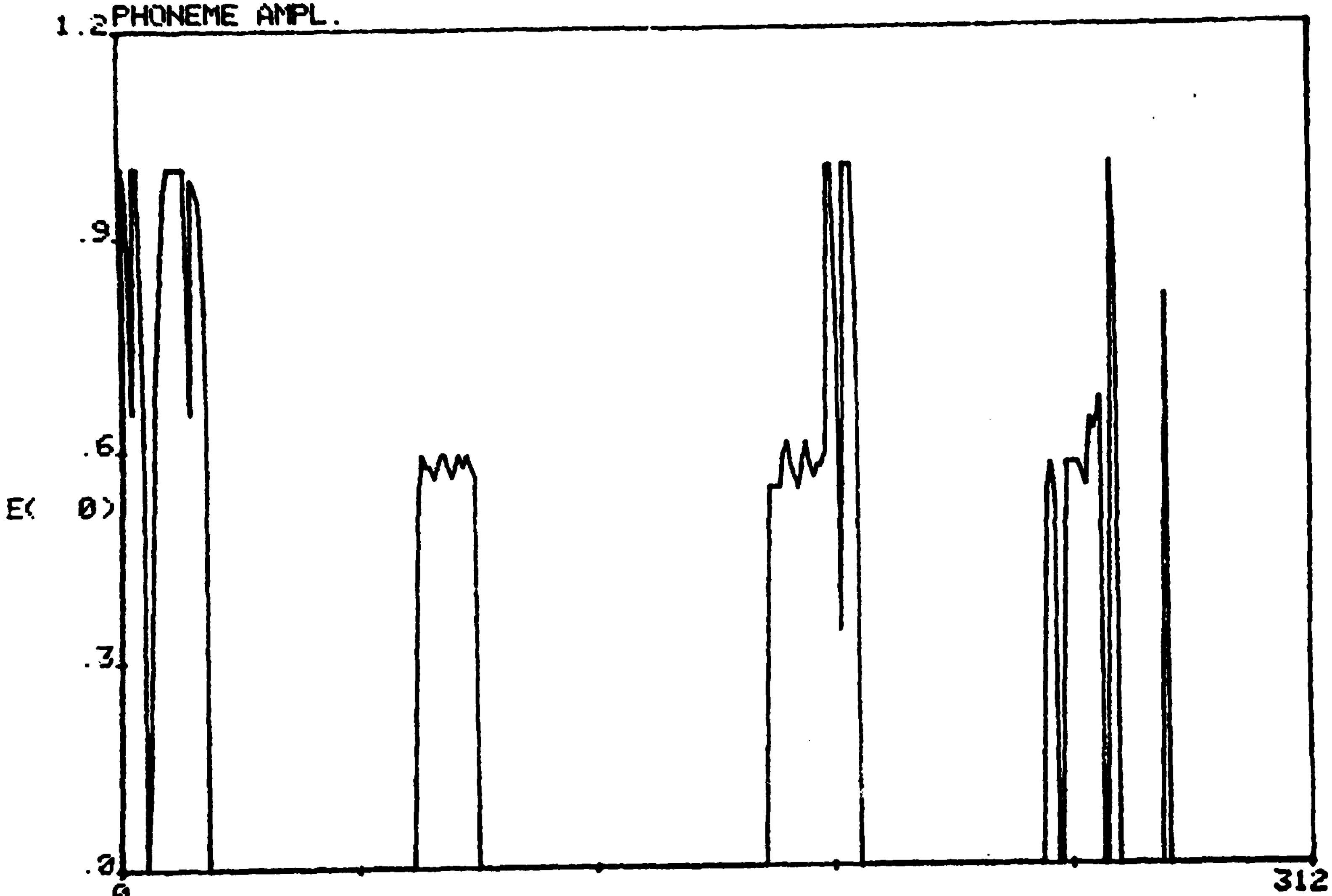

AMPLITUDES FOR PHONEME # 02 FROM FILE 1BTRYSV.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



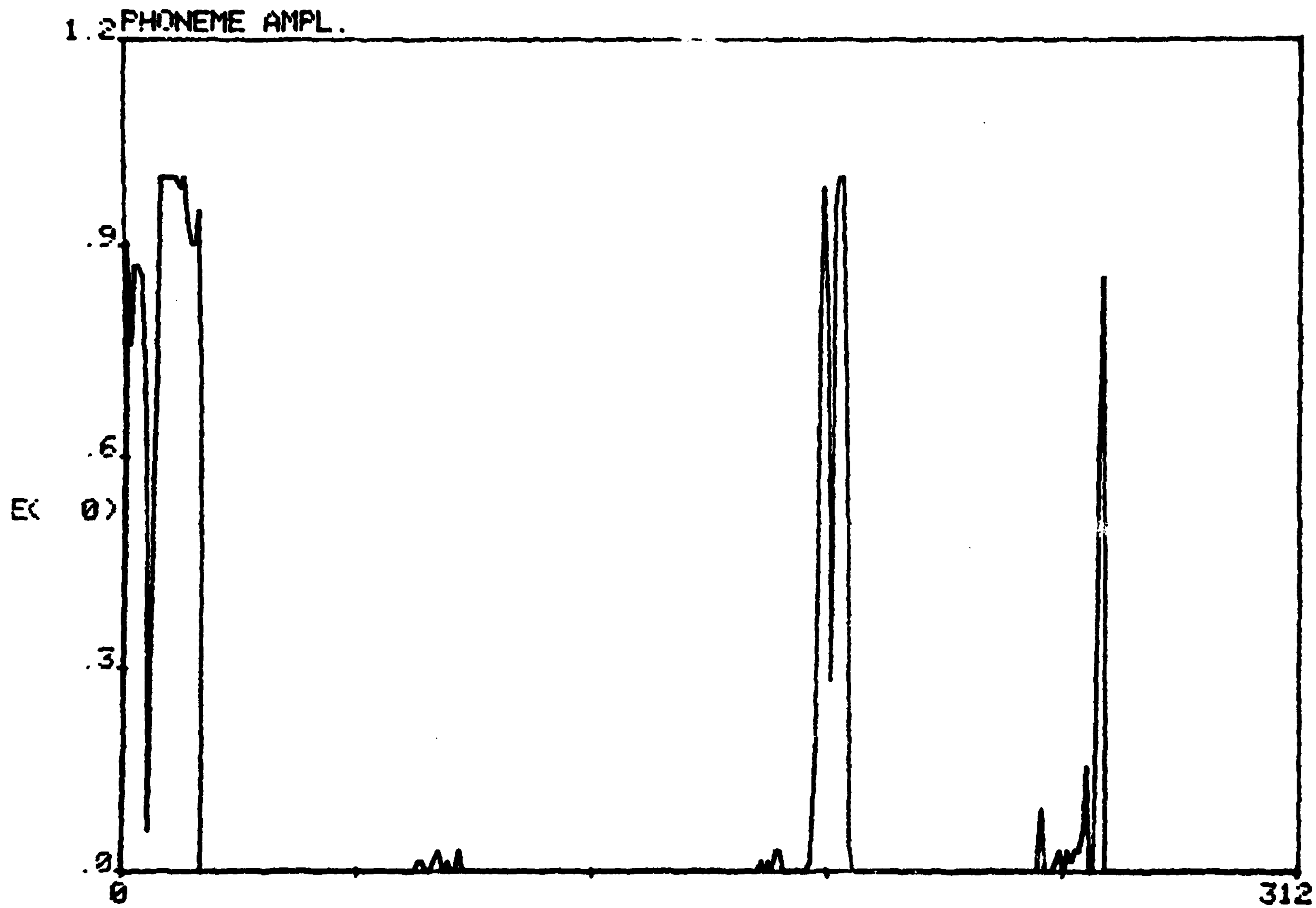

AMPLITUDES FOR PHONEME # 03 FROM FILE 1BTRY1V.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



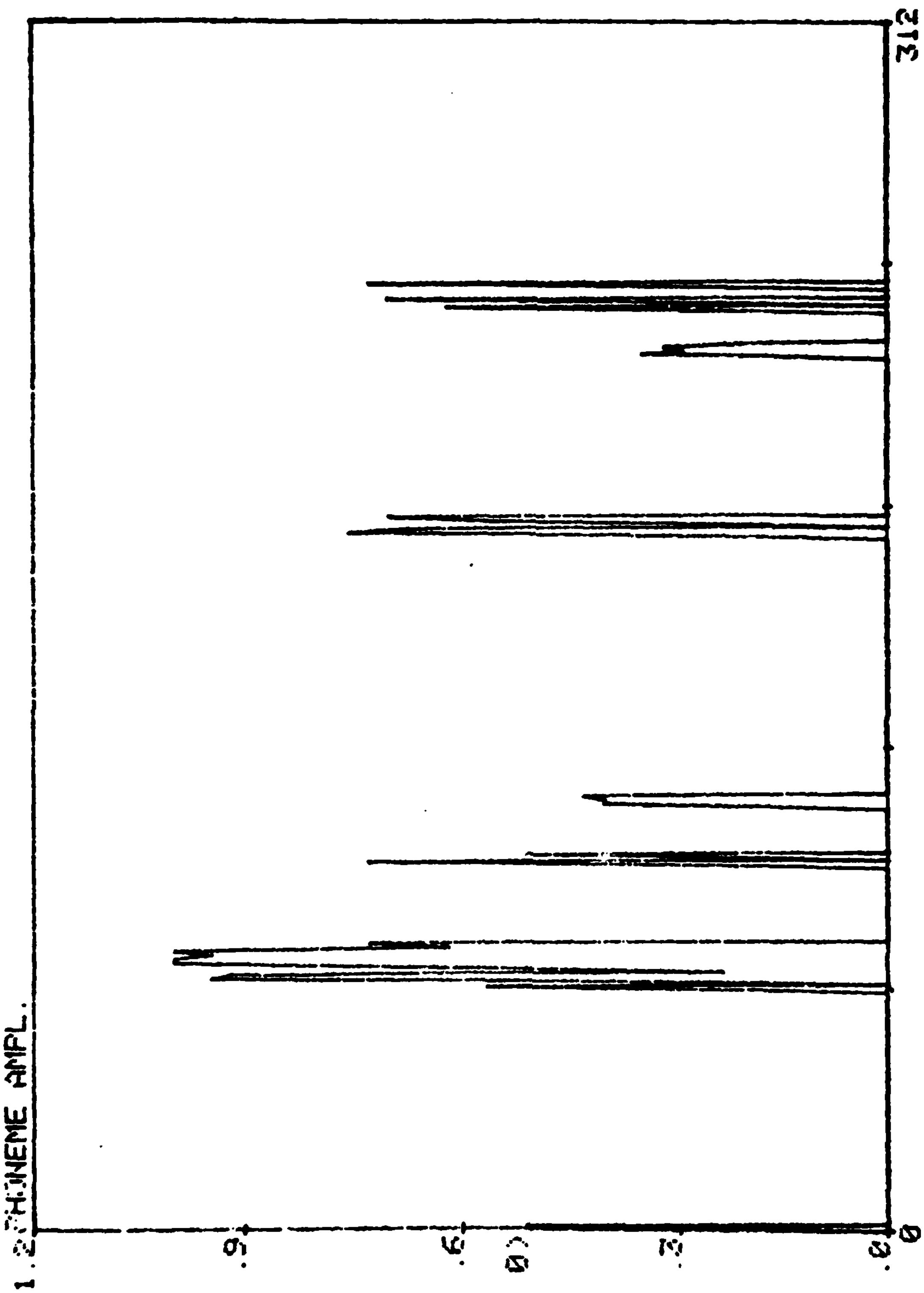
AMPLITUDES FOR PHONEME # 03 FROM FILE 1BTRY2F.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



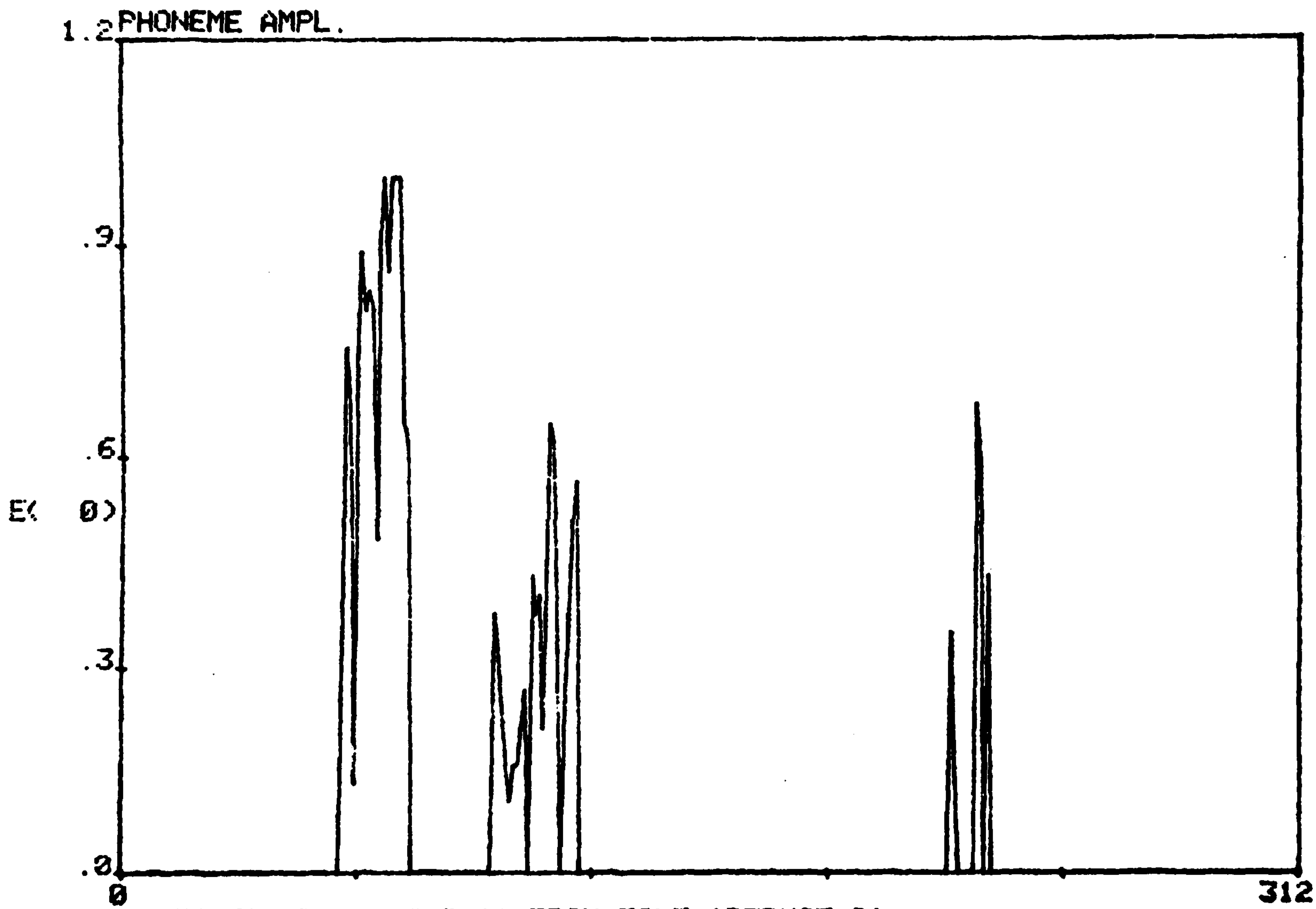
AMPLITUDES FOR PHONEME # 03 FROM FILE 3VFSP1B.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



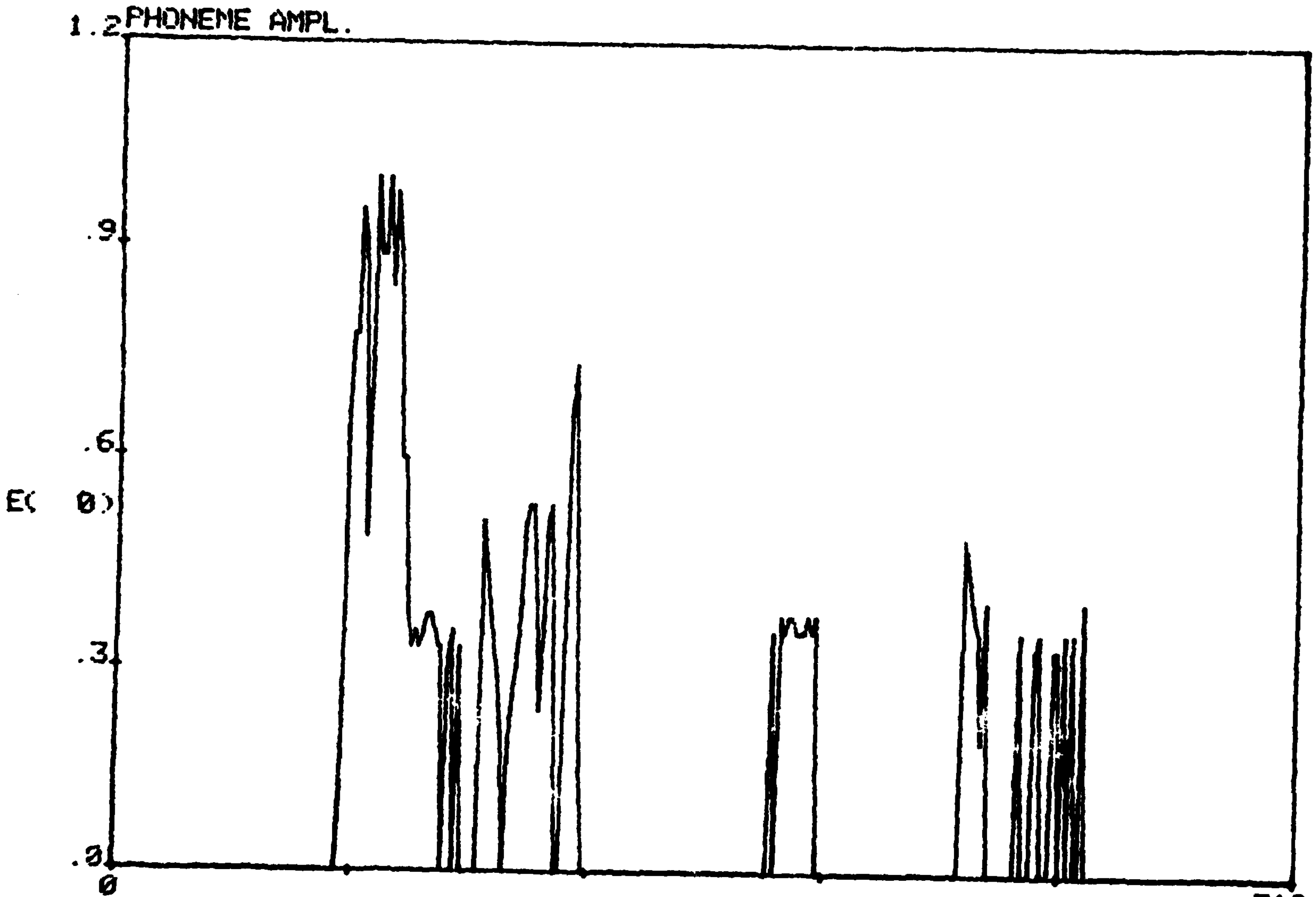
AMPLITUDES FOR PHONEME # 03 FROM FILE 1BTRY5V.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT



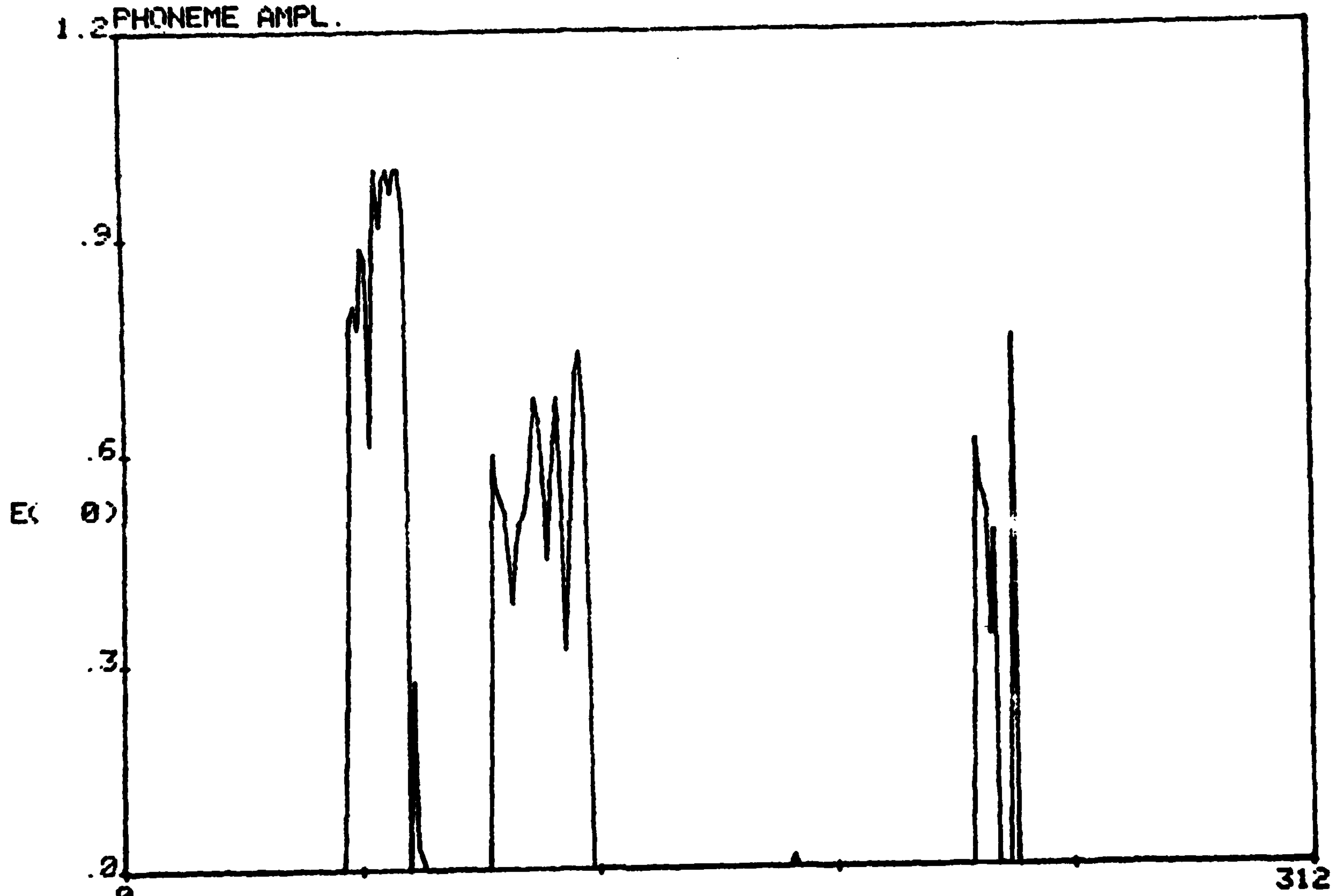

AMPLITUDES FOR PHONEME # 11 FROM FILE 1BTRY1V.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT




AMPLITUDES FOR PHONEME # 11 FROM FILE 1BTRY2F.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT




AMPLITUDES FOR PHONEME # 11 FROM FILE 3VFSP1B.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT




AMPLITUDES FOR PHONEME # 11 FROM FILE 1BTRY5V.01
WORDS SPOKEN AND SPEAKER: 0-1-2-3 BY SEELANDT

## C. Plotting the Spectral Components of a Speechfile

The program RECORD was used to plot the spectral components of any given vector of a speechfile. This was used to compare the results of a 64 point DFT with a 128 point DFT, both performed on the same speechfile. Figure C.1 and C.2 illustrate the resulting plots for a 128 point DFT. Figure C.3 is the corresponding plots for a 64 point DFT. As can be seen, the 64 point DFT has a "smoothing" action associated with it.

## PROGRAM RECORD

| | |
|---|---|
| File: | RECORD |
| Language: | Fortran 5 |
| Date: | November 15, 1982 |
| Author: | J. Fletcher |
| Subject: | Plot of Spectral components of a Speechfile |
| Calling Sequence: | RECORD |

| ARGUMENT | TYPE | PURPOSE |
|---|---|---|
| ICOMP | Integer | number of frequency components |
| IRCC | Integer | sets record size |
| ISTART | Integer | first component read from file |
| IFILE | string | name of spectrum file |
| COMPNT | Integer Array | array for components |

PURPOSE:

This program is used to plot the spectral components of a speechfile.

DESCRIPTION:

Location: FLETCHER. DR

ARGUMENT STRUCTURE:

ICOMP = number of frequency components per vector in integer form. It is of the power of two (i.e. 32,64 or 128).

ISTART = integer number of vector desired to be used. This must be a three digit number

IFILE = The name of the spectrum file to be used. It is in a 7 character Hollerith string format.

PROGRAM USE:

This program uses a subroutine GRPH2 to plot the spectral components of a vector on the Tektronix 4010 terminal. The vector and spectrum file is selected by the user. The user must also instruct the program as to how many spectral components there are per vector. GRPH.LB must be loaded with this program.

```
C       ****************************************
C       **                                    **
C       **    PROGRAM:   R E C O R D          **
C       **                                    **
C       **    BY:   CPT JAMES E. FLETCHER     **
C       **                                    **
C       **    DATE:     15 NOV  82            **
C       **                                    **
C       ****************************************
C
C       NOTE:  THIS PROGRAM IS FOR FORTRAN 5 AND MUST BE RUN ON THE
C              TEKTRONIX 4010 TERMINAL.
C
C
C          This program uses a spectrum file created by the program SGRAM
c       to plot the spectral components for a time slice on the TeKtronix
C       4010 terminal.  The user has the option of choosing his spectrum
C       file, the number of frequency components per time slice (limited
C       to a maximum of 128 spectral components), and the time slice he
C       wishes to observe.
C
C
        DIMENSION IBI(128), COMPNT(128), IFILE(13)
  5     DO 10 I=1,128
        IBI(I) = 0
        COMPNT(I) = 0
  10    CONTINUE
        ACCEPT"HOW MANY FREQ. COMPONENTS (POWER OF 2)?",ICOMP
        IREC = ICOMP * 2     ;  Sets record size when opening spectrum file.
        ACCEPT"VECTOR TO BE PLOTTED (ENTER THREE DIGITS): "
        READ(11,20) ISTART       ; Time slice to be observed.
  20    FORMAT(I3)
        ACCEPT"COMPONENTS FROM SPEECHFILE:"
        READ(11,30) IFILE(1)     ;Name of speechfile spectrum came from.
  30    FORMAT(S13)
        CALL OPEN(2,IFILE,1,IREC,IER1)  ; Open spectrum file
        CALL CHECK(IER1)
        CALL READR(2,ISTART,IBI,1,IER2)     ; Read one record from file.
        CALL CHECK(IER2)
        DO 40 I=1,128    ; Create a real array of components
        COMPNT(I) = FLOAT(IBI(I))     ; Maximum size is 128 components.
  40    CONTINUE
        ACCEPT"PRINT COMPONENTS ON PRINTER? (1=Y/0=N)",IPRNT
        IF(IPRNT .EQ. 0) GO TO 60
        WRITE(12,50) (COMPNT(I), I=1,128)
  50    FORMAT(5X,F10.2,5X,F10.2,5X,F10.2,5X,F10.2)
C
C       GRPH 2 prints a plot of values that are in COMPNT array.
C
  60    CALL GRPH2("VECTOR FREQ. COMPONENTS",1,COMPNT,U,ICOMP,0,XMIN,XMAX,0)
        WRITE(10,70) ISTART, IFILE(1)    ; Header information for graph
  70    FORMAT("PLOT FOR VECTOR ",I3," FROM FILE ",S13)
        READ(11,80) IPAUS
  80    FORMAT(S1)
```

```
CALL RESET
CALL ERS(1)
CALL TDELAY(2)
ACCEPT"RETURN ? (1=Y/0=N) ",IANS ; Return to look at another time slice
IF(IANS .EQ. 1) GO TO 5
STOP
END
```

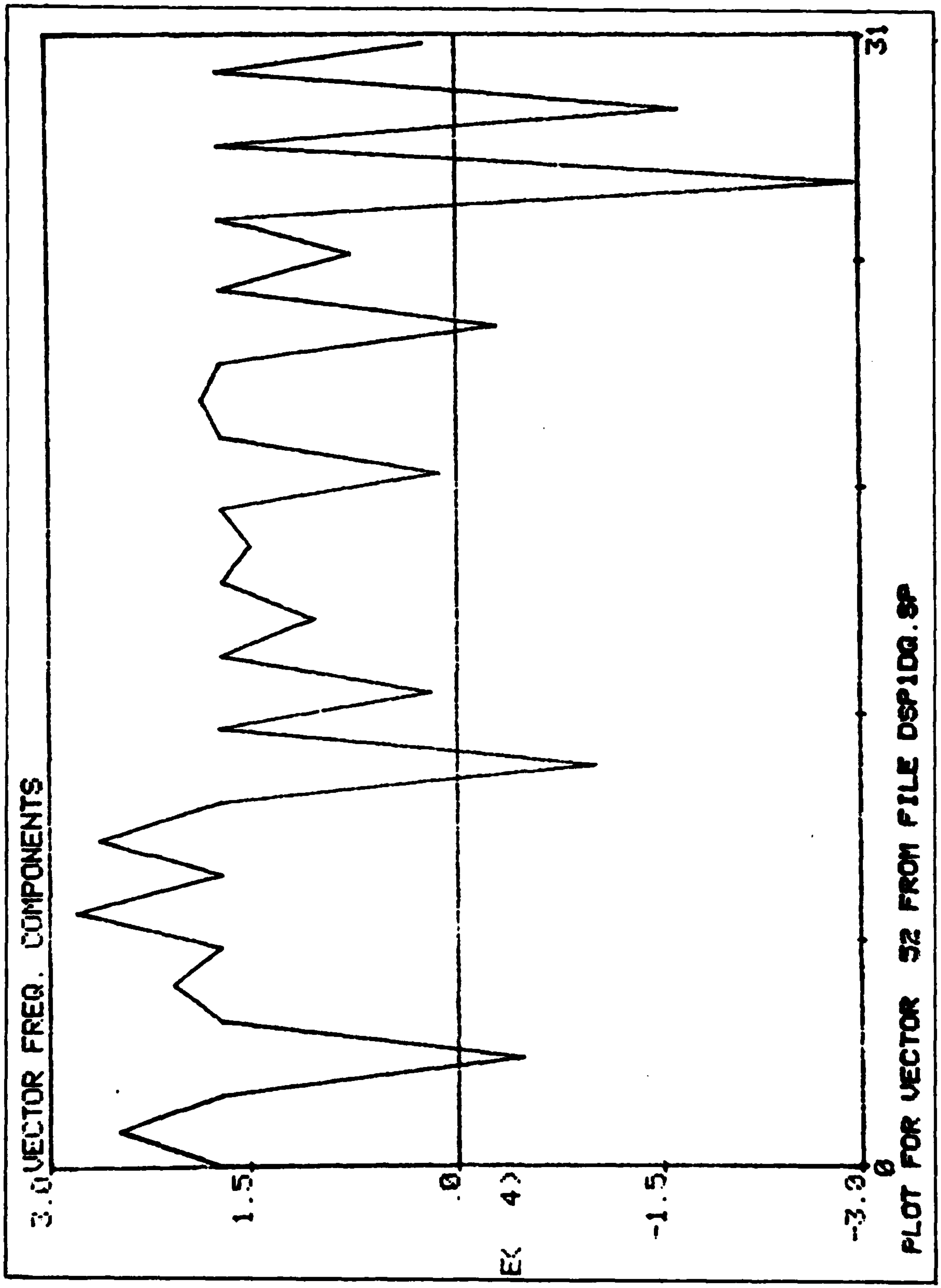



Figure C.1  128 Point DFT

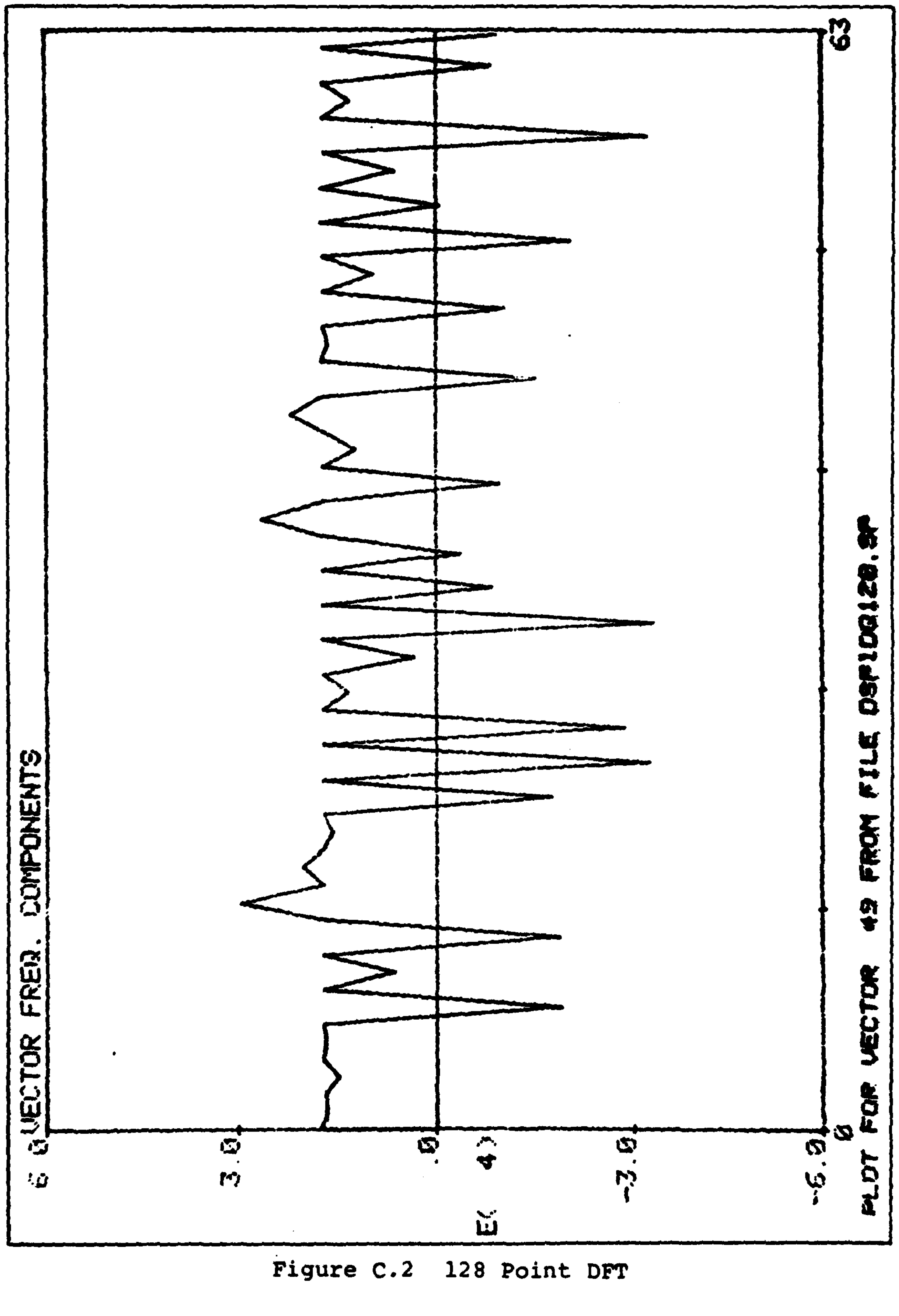



Figure C.2  128 Point DFT

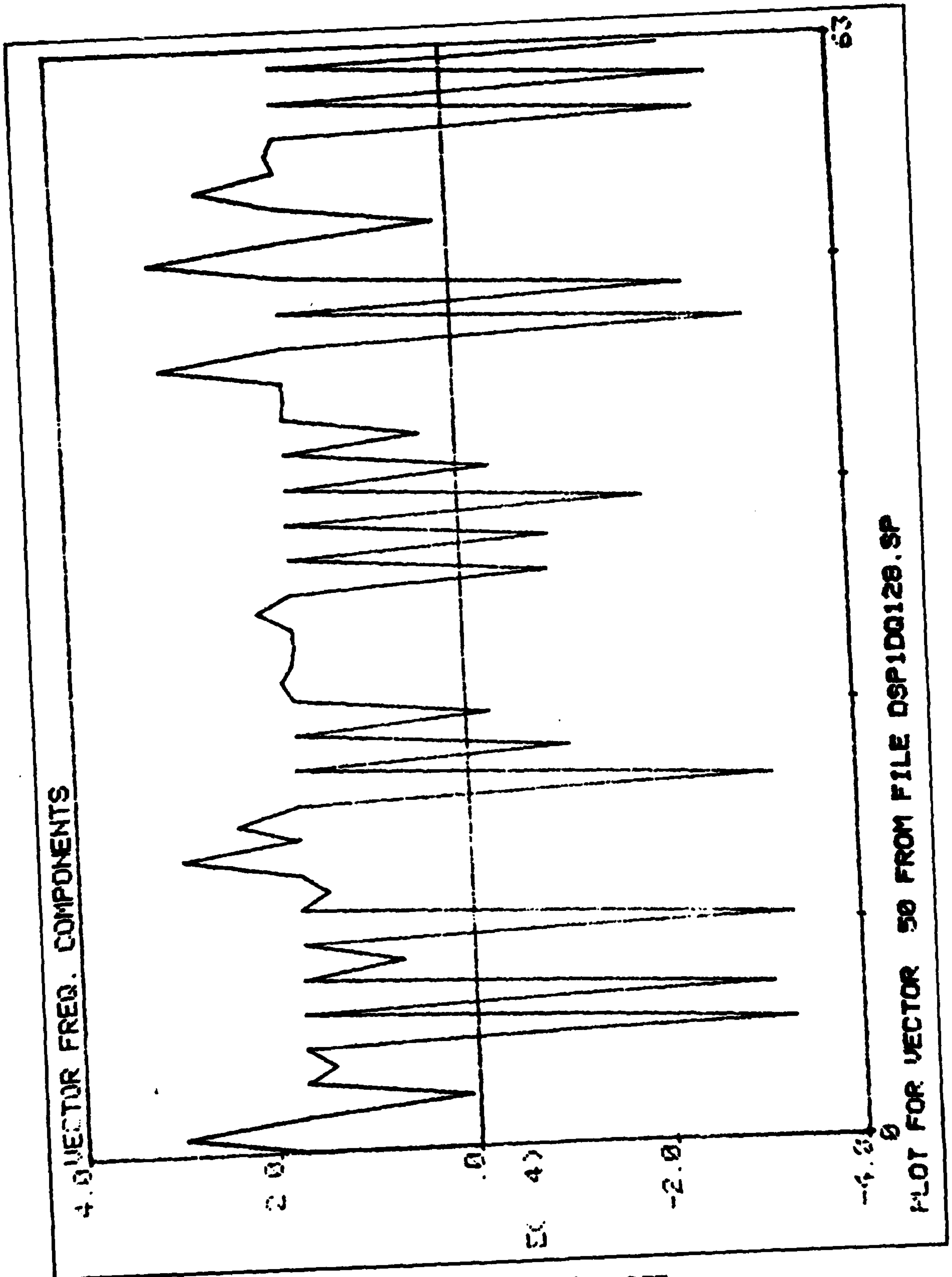



Figure C.3 64 Point DFT

# VITA

Captain James E. Fletcher was born on 6 September 1955 in Philadelphia, Pennsylvania. He graduated from high school in Dalzell, South Carolina, in 1973 and attended the University of South Carolina from which he received the degree of Bachelor of Science in Electrical Engineering in May 1977. Upon graduation, he received a commission in the USAF through the ROTC program. He then attended the Communications - Electronics Engineer course at Keesler AFB in June 1977. He then served as an engineering team chief and as a program director while stationed at HQ SAC, Offutt AFB until entering the School of Engineering, Air Force Institute of Technology, in June 1981.

Permanent address: Route 2, Box 106
Dalzell, South Carolina
29040

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A144 960

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited. |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GE/EE/84J-2 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/ENG | 7a. NAME OF MONITORING ORGANIZATION |
| 6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433 | 7b. ADDRESS (City, State and ZIP Code) |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS (City, State and ZIP Code) | 10. SOURCE OF FUNDING NOS.: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO. |
| 11. TITLE (Include Security Classification): See Box 19 | |

12. PERSONAL AUTHOR(S): James E. Fletcher, B.S., Capt, USAF

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Yr., Mo., Day) | 15. PAGE COUNT |
|---|---|---|---|
| MS Thesis | FROM ________ TO ________ | 1984 March | 176 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB. GR. | |
| | | | |
| | | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Approved for public release: IAW AFR 190-17.
Lynn E. Wolaver 8 Aug 84
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

Title: PERFORMANCE IMPROVEMENTS OF THE PHONEME RECOGNITION ALGORITHM

Thesis Chariman: Larry R. Kizer, Major, USAF

| | |
|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐ | 21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL: Larry R. Kizer, Major, USAF | 22b. TELEPHONE NUMBER (Include Area Code); 22c. OFFICE SYMBOL: AFIT/ENG |

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

An analysis of speech is made by comparing single, two and three time slice phonemes to five time slice phonemes (.08 sec/time slice). All phonemes were created from the same speech with the single, two and three time slice phonemes created from portions of the five time slice phonemes. It was found that the single time slice phonemes compared favorably with the five time slice phonemes in recognizing speech only if the frequency components remained relatively constant over a period of time, such as the vowel and nasal sounds. The two time slice phonemes showed results that began to duplicate those of the five time slice phonemes, but still had inconsistant results identifying fricative sounds. Three time slice phonemes results showed a closer correlation with the results of the five time slice phonemes than those of the one and two time slice phonemes. All results were obtained using a 64 sampled, Hamming windowed, Discrete Fourier Transform. The recognition results for each time slice of speech, using various length phonemes, are tabulated and the results are used to re-synthesize the original speech. This was done by using digitized speech composed of the middle time slices from the 71 five time slice phonemes. Results indicated that the synthesized speech was understandable when the recognition results successfully identified the proper phoneme for approximately 4 consecutive time slices. An extraneous phoneme choice in a consecutive grouping of a phoneme choice did not seriously degrade the output since it accounted for only an .08 second time slice.

END

FILMED

9-84

DTIC